IP2022 Wireless Network Processor

Features and Performance Optimized for Network Connectivity

1.0 Product Highlights

The Ubicom IP2022™ Wireless Network Processor combines support for communication physical layer, Internet protocol stack, device-specific application, and device-specific peripheral software modules in a single chip, and is reconfigurable over the Internet. It can be programmed, and reprogrammed, using pre-built software modules and configuration tools to create true single-chip solutions for a wide range of device-to-device and device-to-human communication applications. High speed communication interfaces are available via on-chip hardware Serializer/Deserializer (SerDes) blocks. These full-duplex blocks allow the IP2022 to be used in a variety of communication bridging applications. Each SerDes block is capable of supporting 10Base-T Ethernet (MAC and PHY), USB, GPSI, SPI, or UART. The high-speed operating frequency, combined with most instructions executing in a single cycle, delivers the throughput needed for emerging network connectivity applications. A flash-based program memory allows both in-system and runtime reprogramming. The IP2022 implements most peripheral, communications and control functions via software modules (ipModule™ software), replacing traditional hardware for maximum system design flexibility. This approach allows rapid, inexpensive product design and, when needed, quick and easy reconfiguration to accommodate changes in market needs or industry standards.

Key Features:

- Designed to support single-chip networked solutions
  - Fast processor core
  - 64KB (32K x 16) Flash program memory
  - 16KB (8K x 16) SRAM data/program memory
  - 4KB (4K x 8) SRAM data memory
  - Two SerDes communication blocks supporting common PHYs (Ethernet, USB, UARTs, etc.) and bridging applications
- Advanced 120MHz RISC processor
  - High speed packet processing
  - Instruction set optimized for communication functions
  - Supports software implementation of traditional hardware functions
- In-system reprogrammable for highest flexibility
  - Run time self-programmable
  - Vpp = Vcc supply voltage

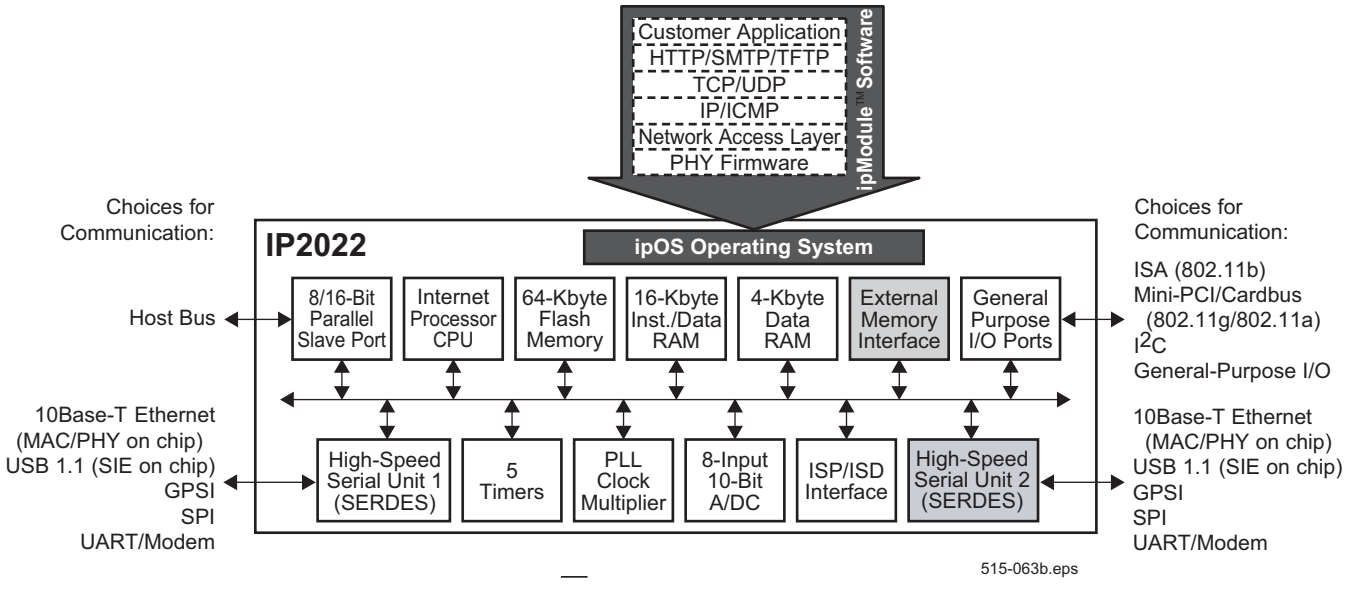


Figure 1-1 IP2022 Block Diagram

© 2001-2008 Ubicom, Inc. All rights reserved.

1.0 Product Highlights 1
1.1 Additional Features . . . 3
1.2 Architecture . . . 4
1.2.1 CPU . . . 4
1.2.2 Serializer/Deserializers . . . 4
1.2.3 Low-Power Support . . . 4
1.2.4 Memory . . . 4
1.2.5 Instruction Set . . . 4
1.2.6 Other Supported Functions . . . 5
1.2.7 Programming and Debugging Support . . . 5

2.0 Pin Definitions 6
2.1 PQFP (Plastic Quad Flat Package) for IP2022 . . . 6
2.2 Signal Descriptions — IP2022 . . . 7

3.0 System Architecture 10
3.1 CPU Registers . . . 11
3.2 Data Memory . . . 13
3.3 Program Memory . . . 13
3.3.1 Loading the Program RAM . . . 14
3.3.2 Program Counter . . . 14
3.4 Low Power Support . . . 14
3.4.1 Clock Stop Mode (SLEEP) . . . 15
3.4.2 Wakeup . . . 15
3.5 Speed Change . . . 15
3.6 Instruction Timing . . . 16
3.7 Interrupt Support . . . 16
3.7.1 Interrupt Processing . . . 16
3.7.2 Global Interrupt Enable Bit . . . 19
3.7.3 Interrupt Latency . . . 19
3.7.4 Return From Interrupt . . . 19
3.7.5 Disabled Interrupt Resources . . . 20
3.8 Reset . . . 20
3.8.1 Brown-Out Detector . . . 22
3.8.2 Reset and Interrupt Vectors . . . 22
3.8.3 Register States Following Reset . . . 22
3.9 Clock Circuitry . . . 23
3.9.1 External Clock Connections . . . 23
3.10 Configuration Block . . . 24
3.10.1 FUSE0 Register (not run-time programmable) . . . 25
3.10.2 FUSE1 Register (not run-time programmable) . . . 26
3.10.3 TRIM0 Register . . . 27

4.0 Instruction Set Architecture 28
4.1 Addressing Modes . . . 28
4.1.1 Pointer Registers . . . 28
4.1.2 Direct Addressing Mode . . . 29
4.1.3 Indirect Addressing Mode . . . 29
4.1.4 Indirect-with-Offset Addressing Mode . . . 30
4.2 Instruction Set . . . 31
4.2.1 Instruction Formats . . . 31
4.2.2 Instruction Types . . . 31
4.3 Instruction Pipeline . . . 33
4.4 Subroutine Call/Return Stack . . . 34
4.5 Key to Abbreviations and Symbols . . . 35
4.6 Instruction Set Summary Tables . . . 35
4.7 Self-Programming and Read Instructions . . . 40
4.7.1 Flash Timing Control . . . 41
4.7.2 Interrupts During Flash Operations . . . 41

5.0 Peripherals 42
5.1 I/O Ports . . . 42
5.1.1 Port B Interrupts . . . 42
5.1.2 Reading and Writing the Ports . . . 43
5.1.3 RxIN Registers . . . 43
5.1.4 RxOUT Registers . . . 43
5.1.5 RxDIR Registers . . . 43
5.1.6 INTED Register . . . 44
5.1.7 INTF Register . . . 44
5.1.8 INTE Register . . . 44
5.1.9 Port Configuration Upon Power-Up . . . 44
5.2 Timer 0 . . . 44
5.3 Real-Time Timer (RTTMR) . . . 45
5.4 Multi-Function Timers (T1 and T2) . . . 47
5.4.1 Timers T1, T2 Operating Modes . . . 47
5.4.2 T1 and T2 Timer Pin Assignments . . . 49
5.4.3 T1 and T2 Timer Registers . . . 49
5.5 Watchdog Timer (WDT) . . . 50
5.6 Serializer/Deserializer (SERDES) . . . 51
5.6.1 SERDES TX/RX Buffers . . . 51
5.6.2 SERDES Configuration . . . 51
5.6.3 SERDES Interrupts . . . 51
5.6.4 Protocol Modes . . . 53
5.6.5 10base-T Ethernet . . . 54
5.6.6 USB . . . 57
5.6.7 UART . . . 60
5.6.8 SPI . . . 61
5.6.9 GPSI . . . 63
5.7 Analog to Digital Converter (ADC) . . . 65
5.7.1 ADC Reference Voltage . . . 65
5.7.2 A/D Converter Registers . . . 65
5.7.3 Using the A/D Converter . . . 66
5.7.4 ADC Result Justification . . . 66
5.8 Comparator . . . 66
5.8.1 CMPCFG Register . . . 66
5.9 Linear Feedback Shift Register (LFSR) . . . 67
5.9.1 LFSRCFG1 Register . . . 68
5.9.2 LFSRCFG2 Register . . . 69
5.9.3 LFSRCFG3 Register . . . 69
5.9.4 DATAIN Register . . . 69
5.9.5 DATAOUT Register . . . 69
5.9.6 DOUT Register . . . 69
5.9.7 FBx Registers . . . 69
5.9.8 POLYx Registers . . . 69
5.9.9 RESx Registers . . . 69
5.9.10 RESCMPx Registers . . . 70
5.9.11 LFSR Configuration . . . 71
5.10 Parallel Slave Peripheral (PSP) . . . 72
5.10.1 PSPCFG Register . . . 72
5.11 External Memory Interface . . . 73
5.11.1 EMCFG Register . . . 73

6.0 In-System Programming 76

7.0 Memory Reference 77
7.0.1 Registers (sorted by address) . . . 77
7.0.2 Program Memory . . . 81
7.1 Register Bit Definitions . . . 82
7.1.1 ADCCFG Register . . . 82
7.1.2 ADCTMR Register . . . 82
7.1.3 CMPCFG Register . . . 82
7.1.4 EMCFG Register . . . 83
7.1.5 FCFG Register . . . 84
7.1.6 INTSPD Register . . . 86
7.1.7 LFSRA Register . . . 86
7.1.8 PSPCFG Register . . . 87
7.1.9 RTCFG Register . . . 88
7.1.10 SxINTE/SxINTF Register . . . 89
7.1.11 SxMODE Register . . . 90
7.1.12 SxRCFG Register . . . 90
7.1.13 SxRCNT Register . . . 91
7.1.14 SxRSYNC Register . . . 91
7.1.15 SxSMASK Register . . . 92
7.1.16 SxTCFG Register . . . 92
7.1.17 SxTMRH/SxTMRL Register . . . 93
7.1.18 SPDREG Register . . . 93
7.1.19 STATUS Register . . . 94
7.1.20 T0CFG Register . . . 95
7.1.21 TxCFG1H Register . . . 95
7.1.22 TxCFG2H Register . . . 96
7.1.23 TxCFG1L Register . . . 96
7.1.24 TxCFG2L Register . . . 97
7.1.25 TCTRL Register . . . 98
7.1.26 XCFG Register . . . 98

8.0 Electrical Characteristics 99
8.1 Absolute Maximum Ratings . . . 99
8.2 DC Specifications: . . . 100
8.3 AC Specifications: . . . 102
8.4 Comparator DC and AC Specifications . . . 103
8.5 ADC 10-bit Converter DC and AC Specifications . . . 103

9.0 Package Dimensions 104
9.1 PQFP . . . 104

10.0 Part Numbering 105

1.1 Additional Features

Internet Processor Capabilities
Foundation for Highly Flexible Connectivity Solution

- Performance: 120 MIPS @ 120 MHz
- Predictable execution for hard real-time applications
- Fast and deterministic 3-cycle (25ns @120MHz) internal interrupt response
- Hardware save/store of key registers
- Functions implemented via software tightly coupled with hardware assist peripherals

Multiple Networking Protocols and Physical Layer Support Hardware

- Two full-duplex serializer/deserializer (SERDES) channels
  - Flexible to support 10Base-T, GPSI, SPI, UART, USB protocols
  - Two channels for protocol bridging
  - On-chip squelch function for 10Base-T Ethernet on each SERDES
- Four hardware LFSR (Linear Feedback Shift Register) units
  - CRC generation/checking
  - Data whitening
  - Encryption

Memory

- 64-Kbyte (32K x 16) on-chip program flash memory
- 16-Kbyte (8K x 16) on-chip program/data RAM
- 4-Kbyte on-chip linear-addressed data RAM
- Self-programming with built-in charge pump: instructions to read, write, and erase flash memory
- Addresses up to 2 Mbytes of external memory

CPU Features

- RISC engine core with DC to 120 MHz operation
  - 8.33 ns instruction cycle at max frequency
- Compact 16-bit fixed-length instructions
- Single-cycle instruction execution on most instructions (3 cycles for jumps and calls)
- Sixteen-level hardware stack for high-performance subroutine linkage
- 8 x 8 signed/unsigned single-cycle multiply
- Pointers and stack operation optimized for C compiler
- Uniform, linear address space (no register banks)

General-Purpose Hardware Peripherals

- Two 16-bit timers with 8-bit prescalers supporting:
  - Timer mode
  - PWM mode
  - Capture/Compare mode
- Parallel host interface, 8/16-bit selectable for use as a communications coprocessor
- External memory interface
- One 8-bit timer with programmable 8-bit prescaler
- One 8-bit real-time clock/counter with programmable 15-bit prescaler and 32 kHz crystal input
- Watchdog timer with prescaler
- 10-bit, 8-channel ADC with 1/2 LSB accuracy
- Analog comparator with hysteresis enable/disable
- Brown-out minimum supply voltage detector
- External interrupt inputs on 8 pins (Port B)

Sophisticated Power and Frequency/Clock Management Support

- Operating voltage of 2.3V to 2.7V
- Switching the system clock frequencies between different clock sources
- On-chip PLL clock multiplier with pre- and post-divider
  - 120 MHz on-chip clock from 4.8 MHz ext. crystal
- Changing the core clock using a selectable divider
- Shutting down the PLL and/or the OSC input
- Dynamic CPU speed control with **`speed`** instruction
- Power-On-Reset (POR) logic

Flexible I/O

- 52 I/O Pins
- 2.3V to 3.6V symmetric CMOS output drive
- 5V-tolerant inputs
- Port A pins capable of sourcing/sinking 24 mA
- Optional I/O synchronization to CPU core clock

Re-configurable Over The Internet

- Customer application program updatable
  - Run-time self programming
- On-chip in-system programming interface
- On-chip in-system debugging support interface
- Debugging at full operating speed
- Programming at device supply voltage level
- Real-time emulation, program debugging, and integrated software development environment offered by leading third-party tool vendors

1.2 Architecture

1.2.1 CPU

The IP2022 implements an enhanced Harvard architecture (i.e. separate instruction and data memories) with independent address and data buses. The 16-bit program memory and 8-bit dual-port data memory allow instruction fetch and data operations to occur in parallel. The advantage of this architecture is that instruction fetch and memory transfers can be overlapped by a multistage pipeline, so that the next instruction can be fetched from program memory while the current instruction is executed with data from the data memory.

Ubicom has developed a revolutionary RISC-based architecture that is deterministic, jitter free, and completely reprogrammable.

The architecture implements a four-stage pipeline (fetch, decode, execute, and write back).

1.2.2 Serializer/Deserializers

One of the key elements in optimizing the IP2022 for device-to-device and device-to-human communication is the inclusion of on-chip serializer/deserializer units. Each unit supports popular communication protocols such as GPSI, SPI, UART, USB, and 10Base-T Ethernet, allowing the IP2022 to be used in bridge, access point and gateway applications.

By performing data serialization and deserialization in hardware, the CPU bandwidth needed to support serial communications is greatly reduced, especially at high baud rates. Providing two units allows easy implementation of protocol conversion or bridging functions between two fast serial devices, such as USB-to-Ethernet, GPSI to ethernet, or Ethernet to Ethernet. A single SerDes unit provides the ability to bridge RS232, SPI, or WLAN (802.11b) to Ethernet.

1.2.3 Low-Power Support

Particular attention has been paid to minimizing power consumption. For example, an on-chip PLL allows use of a lower-frequency external source (e.g., an inexpensive 4.8MHz crystal can be used to produce a 120 MHz on-chip clock), which reduces both power consumption and EMI. In addition, software can change the execution speed of the CPU to reduce power consumption, and a mechanism is provided for automatically changing the speed on entry and return from an interrupt service routine. The **`speed`** instruction specifies power-saving modes that include a clock divisor between 1 and 128. This divisor only affects the clock to the CPU core, not the timers. The **`speed`** instruction also specifies the clock source (OSC1 clock, RTCLK oscillator, or PLL clock multiplier), and whether to disable the OSC1 clock oscillator or the PLL. The **`speed`** instruction executes using the current clock divisor.

1.2.4 Memory

The IP2022 CPU executes from a 32K x 16 flash program memory and an 8K x 16 RAM program/data memory. In addition, the ability to write into the program flash memory allows flexible non-volatile data storage. An interface is available for up to 128K bytes of linearly addressed external memory, which can be expanded to 2M bytes with additional software-based I/O addressing. At 120 MHz operation, the maximum execution rate is 40 MIPS from flash memory and 120 MIPS from RAM. Speed-critical routines can be copied from the flash memory to the RAM for faster execution. The IP2022 has a mechanism for in-system programming of their flash and RAM program memories through a four-wire SPI interface, and software has the ability to reprogram the program memories at run time. This allows the functionality of a device to be changed in the field over the Internet.

1.2.5 Instruction Set

The IP2022 instruction set, using 16-bit words, implements a rich set of arithmetic and logical operations, including signed and unsigned 8-bit x 8-bit integer multiply with a 16-bit product.

1.2.6 Other Supported Functions

On-chip dedicated hardware also includes a PLL, an 8-channel 10-bit ADC, general-purpose timers, single-cycle multiplier, analog comparator, LFSR units, external memory interface, parallel slave port, brown-out power voltage detector, watchdog timer, low-power support, multi-source wakeup capability, user-selectable clock modes, high-current outputs, and 52 general-purpose I/O pins.

1.2.7 Programming and Debugging Support

The IP2022 has advanced in-system programming and debug support on-chip. This unobtrusive capability is provided through the ISP/ISD interface. There is no need for a bond-out chip for software development. This eliminates concerns about differences in electrical characteristics between a bond-out chip and the actual chip used in the target application. Designers can test and revise code on the same part used in the actual application.

Ubicom provides the complete Red Hat GNUPro tools, including C compiler, assembler, linker, utilities and GNU debugger. In addition, Ubicom offers an integrated graphical development environment which includes an editor, project manager, graphical user interface for the GNU debugger, device programmer, and ipModule™ configuration tool.

2.0 Pin Definitions

2.1 PQFP (Plastic Quad Flat Package) for IP2022

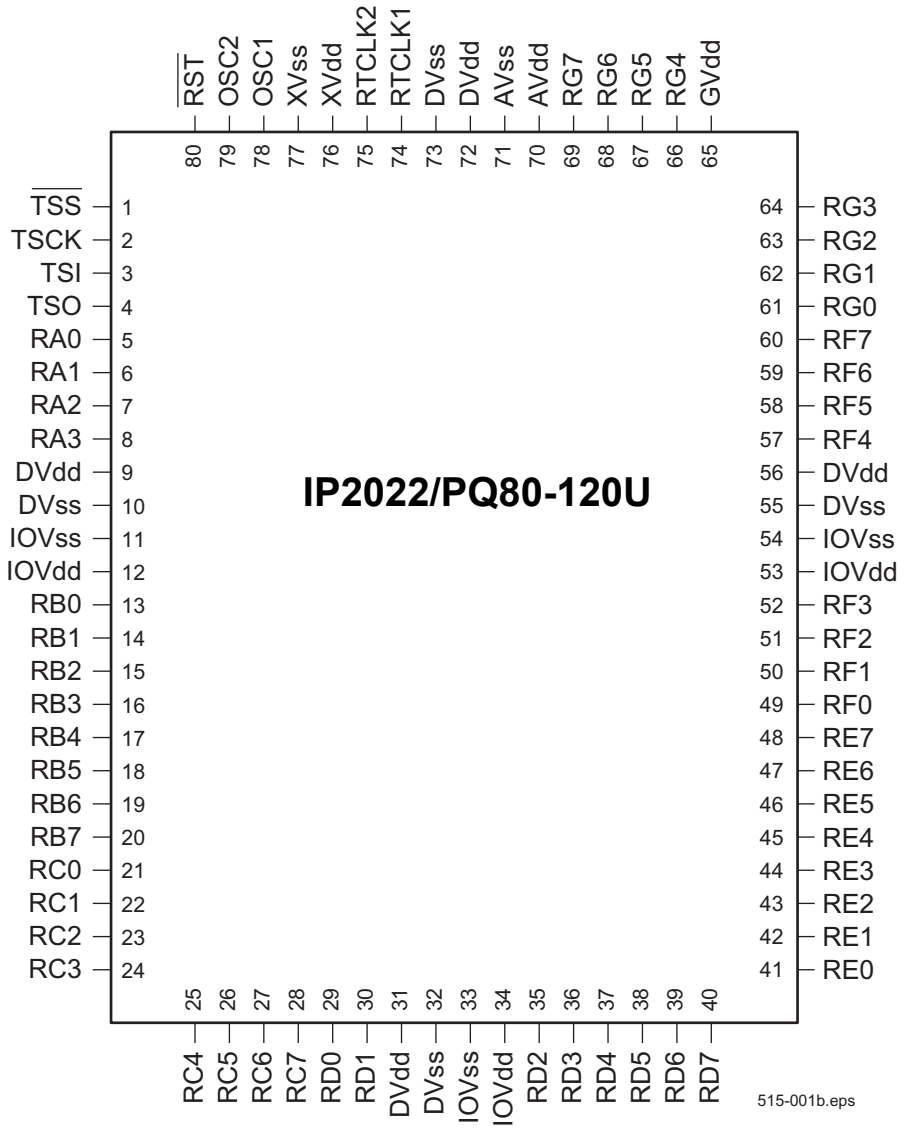


Figure 2-1 IP2022 PQFP Pin Definition (Top View)

2.2 Signal Descriptions — IP2022

I = Digital Input, AI = Analog Input, O/DO = Digital Output, HiZ = High Impedance, P = Power, PLP = On-Chip Pullup, ST = Schmitt Trigger

Table 2-1 Signal Descriptions

Name	Pin	Type	Sink @ 3.3V IOVDD	Source @ 3.3V IOVDD	Function
AVDD	70	P			Analog Supply
AVSS	71	P			Analog Ground
DVDD	9, 31, 56, 72	P			Logic Supply
DVSS	10, 32, 55, 73	P			Logic Ground
GVDD	65	P			I/O Port G supply
IOVDD	12, 34, 53	P			I/O Supply (except Port G)
IOVSS	11, 33, 54	P			I/O Ground (all ports)
XVDD	76	P			PLL Supply
XVSS	77	P			PLL Ground
OSC1	78	I/ST			Clock/Crystal Input
OSC2	79	O/HiZ			Crystal Output (tri-state if FUSE0 bit 15 = 1)
$\overline{RST}$	80	I/ST/ PLP			Reset Input. There is a weak pull-up on this pin, but floating this pin does not guarantee Vih.
RTCLK1	74	I			Real-Time Clock/Crystal Input
RTCLK2	75	O/HiZ			Real-Time Crystal Output (tri-state if FUSE0 bit 14 = 1)
$\overline{TSS}$	1	I/ST/ PLP			Target SPI Slave Select (used only for in-system programming and debug)
TSCK	2	I/ST/ PLP			Target SPI Clock (used only for in-system programming and debug)
TSI	3	I/ST /PLP			Target SPI Serial Data Input (used only for in-system programming and debug)
TSO	4	O/HiZ			Target SPI Serial Data output (used only for in-system programming and debug; high Z unless $\overline{TSS}$ low)
RA0	5	I/O	24 mA	24 mA	I/O Port, High Power Output, Timer 1 Capture 1 Input
RA1	6	I/O	24 mA	24 mA	I/O Port, High Power Output, Timer 1 Capture 2 Input
RA2	7	I/O	24 mA	24 mA	I/O Port, High Power Output, Timer 1 Clock Input
RA3	8	I/O	24 mA	24 mA	I/O Port, High Power Output, Timer 1 Output
RB0	13	I/O	8 mA	8 mA	I/O Port, External Interrupt, Timer 2 Capture 1 Input
RB1	14	I/O	8 mA	8 mA	I/O Port, External Interrupt, Timer 2 Capture 2 Input
RB2	15	I/O	8 mA	8 mA	I/O Port, External Interrupt, Timer 2 Clock Input

Table 2-1 Signal Descriptions (continued)

Name	Pin	Type	Sink @ 3.3V IOVDD	Source @ 3.3V IOVDD	Function
RB3	16	I/O	8 mA	8 mA	I/O Port, External Interrupt, Timer 2 Output
RB4	17	I/O	8 mA	8 mA	I/O Port, External Interrupt, External Memory $\overline{WR}$
RB5	18	I/O	8 mA	8 mA	I/O Port, External Interrupt, Parallel Slave Peripheral $\overline{HOLD}$, External Memory $\overline{RD}$
RB6	19	I/O	8 mA	8 mA	I/O Port, External Interrupt, Parallel Slave Peripheral R/W, External Memory $\overline{LE}$
RB7	20	I/O	8 mA	8 mA	I/O Port, External Interrupt, Parallel Slave Peripheral $\overline{CS}$, External Memory A0
RC0	21	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data D8, External Memory A9
RC1	22	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data D9, External Memory A10
RC2	23	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data D10, External Memory A11
RC3	24	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data D11, External Memory A12
RC4	25	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data D12, External Memory A13
RC5	26	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data D13, External Memory A14
RC6	27	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data D14, External Memory A15
RC7	28	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data D15, External Memory A16
RD0	29	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data D0, External Memory shared A1/D0
RD1	30	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data D1, External Memory shared A2/D1
RD2	35	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data D2, External Memory shared A3/D2
RD3	36	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data D3, External Memory shared A4/D3
RD4	37	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data D4, External Memory shared A5/D4
RD5	38	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data D5, External Memory shared A6/D5
RD6	39	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data D6, External Memory shared A7/D6
RD7	40	I/O	4 mA	4 mA	I/O Port, Parallel Slave Peripheral Data D7, External Memory shared A8/D7
RE0	41	I/O	8 mA	8 mA	I/O Port, S1CLK - SCLK (SPI), RxCLK (GPSI), optional SERDES clock input for UART or USB.

Table 2-1 Signal Descriptions (continued)

Name	Pin	Type	Sink @ 3.3V IOVDD	Source @ 3.3V IOVDD	Function
RE1	42	I/O	8 mA	8 mA	I/O Port, S1RXP - VP (USB), SS (SPI Slave), TxEN (GPSI Master), RxEN (GPSI Slave)
RE2	43	I/O	8 mA	8 mA	I/O Port, S1RXM - VM (USB)
RE3	44	I/O	8 mA	8 mA	I/O Port, S1RXD - RCV (USB), RXD (UART), DI (SPI), TxD (GPSI Master), RxD (GPSI Slave)
RE4	45	I/O	8 mA	8 mA	I/O Port, S1TXPE/S1OE - TxD+ (Ethernet), OE (USB), RxEN (GPSI Master), TxEN (GPSI Slave)
RE5	46	I/O	24 mA	24 mA	I/O Port, High Power Output, S1TXP - Tx+ (Ethernet), VPO (USB), TXD (UART), DO (SPI), RxD (GPSI Master), TxD (GPSI Slave)
RE6	47	I/O	24 mA	24 mA	I/O Port, High Power Output, S1TXM - Tx- (Ethernet), VMO (USB), TxCLK/RxCLK (GPSI Master), TxCLK (GPSI Slave)
RE7	48	I/O	8 mA	8 mA	I/O Port, S1TXME - TxD- (Ethernet), TxBUSY (GPSI)
RF0	49	I/O	8 mA	8 mA	I/O Port, S2TXPE/S2OE - TxD+ (Ethernet), OE (USB), RxEN (GPSI Master), TxEN (GPSI Slave)
RF1	50	I/O	24 mA	24 mA	I/O Port, High Power Output, S2TXP - Tx+ (Ethernet), VPO (USB), TXD (UART), DO (SPI), RxD (GPSI Master), TxD (GPSI Slave)
RF2	51	I/O	24 mA	24 mA	I/O Port, High Power Output, S2TXM - Tx- (Ethernet), VMO (USB), TxCLK/RxCLK (GPSI Master), TxCLK (GPSI Slave)
RF3	52	I/O	8 mA	8 mA	I/O Port, S2TXME - TxD- (Ethernet), TxBUSY (GPSI)
RF4	57	I/O	8 mA	8 mA	I/O Port, S2CLK - SCLK (SPI), RxCLK (GPSI), optional SERDES clock input for UART or USB.
RF5	58	I/O	8 mA	8 mA	I/O Port, S2RXP - VP (USB), SS (SPI Slave), TxEN (GPSI Master), RxEN (GPSI Slave)
RF6	59	I/O	8 mA	8 mA	I/O Port, S2RXM - VM (USB)
RF7	60	I/O	8 mA	8 mA	I/O Port, S2RXD - RCV (USB), RXD (UART), DI (SPI), TxD (GPSI Master), RxD (GPSI Slave)
RG0	61	AI/DO	4 mA*	4 mA*	Output Port, ADC0 Input, Comparator Output
RG1	62	AI/DO	4 mA*	4 mA*	Output Port, ADC1 Input, Comparator – Input
RG2	63	AI/DO	4 mA*	4 mA*	Output Port, ADC2 Input, Comparator + Input
RG3	64	AI/DO	4 mA*	4 mA*	Output Port, ADC3 Input, ADC reference Input
RG4	66	AI/DO	4 mA*	4 mA*	Output Port, ADC4 Input, S1RX-, Squelch 1-
RG5	67	AI/DO	4 mA*	4 mA*	Output Port, ADC5 Input, S1RX+, Squelch 1+
RG6	68	AI/DO	4 mA*	4 mA*	Output Port, ADC6 Input, S2RX-, Squelch 2-
RG7	69	AI/DO	4 mA*	4 mA*	Output Port, ADC7 Input, S2RX+, Squelch 2+

* GVDD = 2.5V

3.0 System Architecture

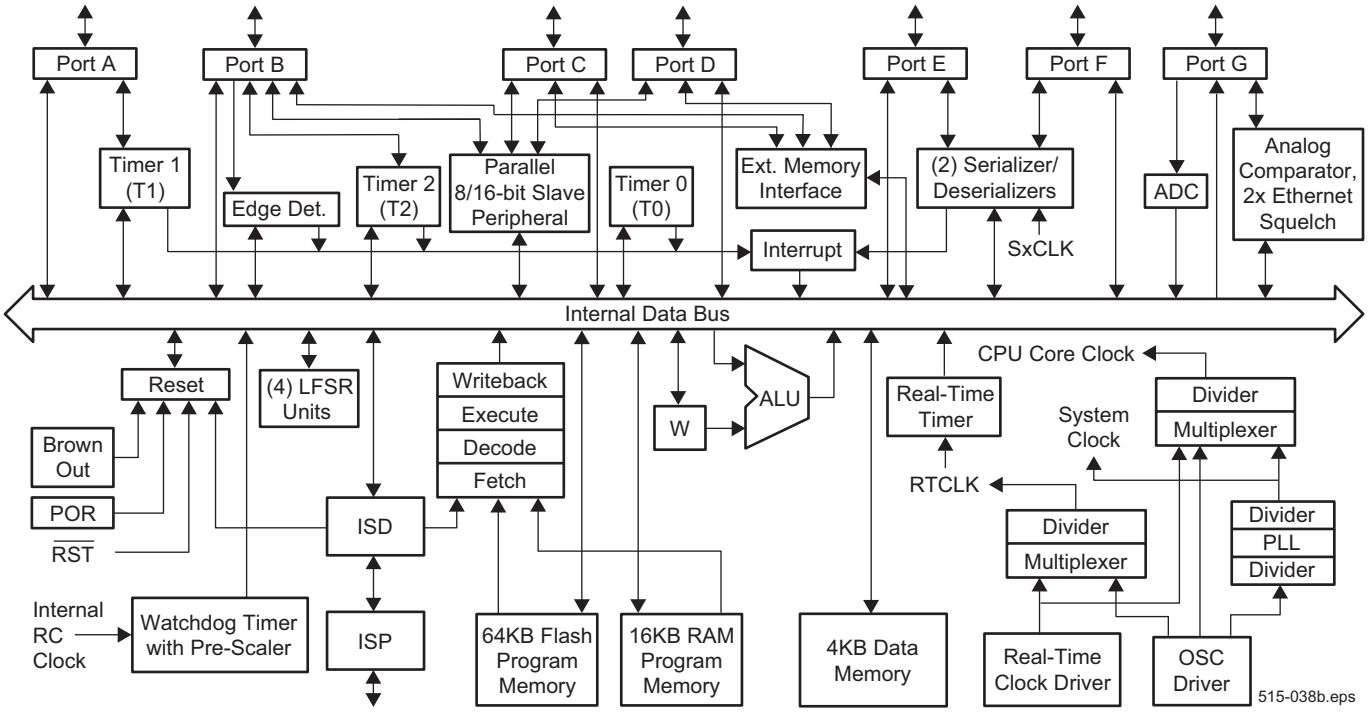


Figure 3-1 IP2022 Detailed Block Diagram

The IP2022 CPU executes from a 32K x 16 bit flash program memory and an 8K x 16 bit RAM program memory. Figure 3-1 shows the IP2022 detailed block diagram. At 120 MHz operation, the maximum execution rate is 40 MIPS from Flash and 120 MIPS from RAM. Speed-critical routines can be copied from the flash memory to the RAM for faster execution.

The CPU operates on 8-bit data in 128 special-purpose registers, 128 global registers, and 3840 bytes of data memory. The special-purpose registers hold control and status bits used for CPU control and for interface with hardware peripherals (timers, I/O ports, A/D converter, etc.)Although the philosophy followed in the design of Ubicom products emphasizes the use of fast RISC CPUs with predictable execution times to emulate peripheral devices in software (the ipModule™ concept), there are a few hardware peripherals which are difficult to emulate in software alone (e.g. an A/D converter) or consume an excessive number of instruction cycles when operating at high speed (e.g. data serialization/deserialization). The design of the IP2022 incorporates only those hardware peripherals which can greatly accelerate or extend the reach of the ipModule™ concept. The hardware peripherals included on-chip are:

- 52 I/O port pins
- Watchdog timer
- Real-time timer
- 2 Multifunction 16-bit timers with compare and capture registers
- 2 Real-time 8-bit timers
- 2 Serializer/Deserializer (SERDES) units
- 4 Linear feedback shift register (LFSR) units
- 10-bit, 8-channel A/D converter
- Analog comparator
- Parallel slave peripheral interface
- External SRAM interface

There is a single interrupt vector which can be reprogrammed by software. On-chip peripherals and up to 8 external inputs can raise interrupts.

There are five sources of reset:

- $\overline{\text{RST}}$ external reset input
- Power-On Reset (POR) logic
- Brown-Out Reset (BOR) logic (detects low AVdd condition)
- Watchdog timer reset
- In-system debugging/programming interface reset

An on-chip PLL clock multiplier (x50) enables high-speed operation (up to 120 MHz) from a slow-speed external clock input or crystal. A CPU clock-throttling mechanism allows fine control over power consumption in modes that

do not require maximum speed, such as waiting for an interrupt.

The IP2022 has a mechanism for in-system programming of its flash and RAM program memories through a four-wire SPI interface. This provides easy programming and reprogramming of devices on assembled circuit boards. In addition, the flash memory can be programmed by software at run time, for example to store user-specific data such as phone numbers and to receive software upgrades downloaded over the Internet. The devices also have an on-chip debugging facility which makes the internal operation of the chip visible to third-party debugging tools.

3.1 CPU Registers

Figure 3-2 shows the CPU registers, which consist of seven 8-bit registers, seven 16-bit registers, and one 24-bit register. The 16-bit registers are formed from pairs of 8-bit registers, and the 24-bit register is formed from three 8-bit registers. For the register quick reference guide, see Section 7.0 and Section 7.1.

The W or working register is used as the source or destination for most arithmetic, movement, and logical instructions.

The STATUS register holds the condition flags for the results of arithmetic and logical operations, the page bits (used for jumps and subroutine calls), and bits which indicate the skipping state of the core and control of continuation skip after return from interrupt. Figure 3-3 shows the assignment of the bits in the STATUS register.

.

7	6	5	4	3	2	1	0
PA2:0			SAR	SSF	Z	DC	C

Figure 3-3 STATUS Register

When writing directly to the STATUS register, it is recommended to use only setb and clrb commands, since writing to the whole STATUS register at once may produce unexpected results. For example, a CLR STATUS command may set the Z bit, and leave the DC and C bits unchanged.

- *PA2:PA0*—Program memory page select bits. Used to extend the 13-bit address encoded in jump and call instructions (selects 8K-word pages). Modified using the **`page`** instruction.
- *SAR*—Skip After Return bit. This bit should be set if the core should be in the skipping state, and should not be set if the core should not be in the skipping state after the completion of the return instruction (**`ret`**, **`retnp`**, or **`retw`** instructions, but not **`reti`**). The return instruction will also clear the SAR control bit to ensure correct behavior after the dynamic jump.
- *SSF*—Shadowed Skipping/not state Flag. Gives the ISR the ability to know if the interrupt occurred immediately following a skip instruction. The software can choose either to clear the SSF flag in the ISR (Interrupt Service Routine) or to make the first instruction of the context switching code a **`nop`** to flush out the skip state.
- *Z*—Zero bit. Affected by most logical, arithmetic, and data movement instructions. Set if the result was zero, otherwise cleared.
- *DC*—Digit Carry bit. After addition, set if carry from bit 3 occurred, otherwise cleared. After subtraction, cleared if borrow from bit 3 occurred, otherwise set.
- *C*—Carry bit. After addition, set if carry from bit 7 of the result occurred, otherwise cleared. After subtraction, cleared if borrow from bit 7 of the result occurred, otherwise set. After rotate (**`rr`** or **`rl`**)

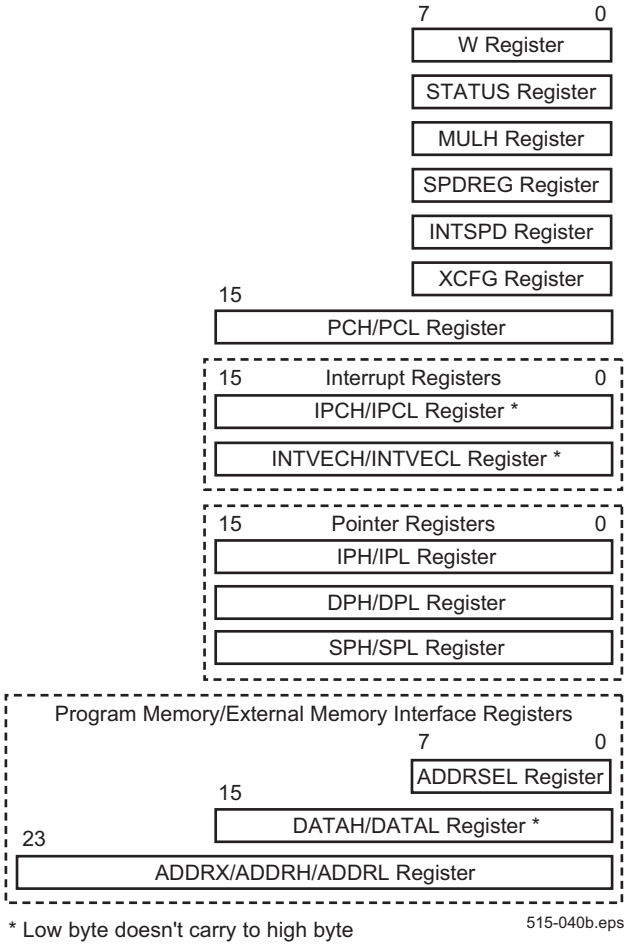


Figure 3-2 CPU Registers

instructions, loaded with the LSB or MSB of the operand, respectively.

The MULH register receives the upper 8 bits of the 16-bit product from signed or unsigned multiplication. The lower 8 bits are loaded into the W register.

The SPDREG register holds bits that control the CPU speed and clock source settings, and is loaded by using the **speed** instruction, as shown in Figure 3-4. The SPDREG register is read-only, and its contents may only be changed by executing a **speed** instruction, taking an interrupt, or returning from an interrupt. For more information about the **speed** instruction and the clock throttling mechanism, see Section 3.4 and Figure 3-16.

Note: The **speed** instruction should be followed by a **nop** instruction if Port B interrupt is used to wake up from sleep mode.

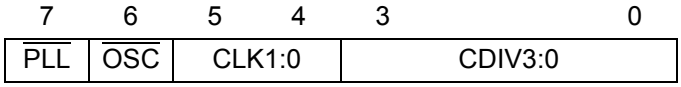


Figure 3-4 SPDREG Register

- $\overline{\text{PLL}}$—enable x50 PLL clock multiplier; 0 = enabled; 1 = disabled. Power consumption can be reduced by disabling it. See Figure 3-16.
- $\overline{\text{OSC}}$—enable OSC oscillator; 0 = enabled; 1 = disabled (stops OSC oscillator and blocks propagation of OSC1 external clock input). Power consumption can be reduced by disabling it.
- CLK1:0—selects the system clock source, as shown in Table 3-1. See Figure 3-16 for the clock logic. See Section 7.1.5 (FCFG register, FRDTS1:0 bits) for exceptions.

Table 3-1 CLK1:0 Field Encoding

CLK1:0	System Clock Source
00	PLL Clock Multiplier
01	OSC Oscillator/External OSC1 Input
10	RTCLK oscillator/external clock on RTCLK1 input
11	System Clock Off

- *CDIV3:0*—selects the clock divisor used to generate the CPU core clock from the system clock, as shown in Table 3-2 (also see Figure 3-16).

Table 3-2 System Clock Divisor

CDIV3:0	System Clock Divisor	CPU Core Frequency
0000	1	120 MHz
0001	2	60 MHz
0010	3	40 MHz
0011	4	30 MHz
0100	5	24 MHz
0101	6	20 MHz
0110	8	15 MHz
0111	10	12 MHz
1000	12	10 MHz
1001	16	7.5 MHz
1010	24	5 MHz
1011	32	3.75 MHz
1100	48	2.5 MHz
1101	64	1.875 MHz
1110	128	0.9375 MHz
1111	Clock Off	0 MHz

The INTSPD register holds bits that control the CPU speed and clock source during interrupt service routines (it is copied to the SPDREG register when an interrupt occurs). It has the same format as the SPDREG register.

When the OSC crystal driver is stopped (SPDREG bit 6 = 1) and Port B or Real Time Timer interrupts are enabled, then INTSPD bits 5 and 4 must not both be 0, because the crystal startup time plus PLL startup time may be greater than WUDP2:0 (see Figure 3-16).

The XCFG register holds additional control and status bits, as shown in Figure 3-5.

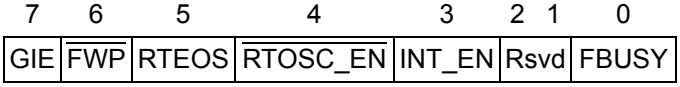


Figure 3-5 XCFG Register

- *GIE*—global interrupt enable bit. When set, interrupts are enabled. When clear, interrupts are disabled. For more information about interrupt processing, see Section 3.7.
- *FWP*—flash write protect bit. When clear, writes to flash memory are ignored. For more information about programming the flash memory, see Section 4.7.

- *RTEOS*—real-time timer oversampling enable bit. When set, oversampling is used. For more information, see Section 5.3.
- *RTOSC_EN*—RTCLK oscillator enable bit. When clear, the RTCLK oscillator is operational. When set, the RTCLK oscillator is turned off.
- *INT_EN*—**int** instruction interrupt enable bit. When set, **int** instructions cause interrupts. When clear, **int** instructions only increment the PC, like **nop**.
- *FBUSY*—read-only flash memory busy bit. Set while fetching instructions out of flash memory or while busy processing an **iread**, **ireadi**, **iwrite**, **iwritei**, **fwrite**, **fread** or **ferase** instruction that operates on Flash, otherwise clear. For more information about programming the flash memory, see Section 4.7.

The PCH and PCL register pair form a 16-bit program counter. The PCH register is read-only. The PCL register can be used to implement a lookup table, by moving a variable to the w register, then executing an **add PCL,w** instruction. If w=01 when the add occurs, the instruction after the add will be skipped; if w=02, two instructions will be skipped, etc.

The IPCH and IPCL register pair specifies the return address when a **reti** instruction is executed.

The INTVECH and INTVECL register pair specifies the interrupt vector. It has a default value of 0 following reset. On a return from interrupt, an option of the **reti** instruction allows software to save the incremented value of the program counter in the INTVECH and INTVECL registers.

The IPH and IPL register pair is used as a pointer for indirect addressing. For more information about indirect addressing, see Section 4.1.3.

The DPH and DPL register pair and the SPH and SPL register pair are used as pointer registers for indirect-with-offset addressing. For more information about indirect-with-offset addressing, see Section 4.1.4. The SPH and SPL registers are automatically post-decremented when storing to memory with a **push** instruction, and they are automatically pre-incremented when reading from memory with a **pop** instruction.

The ADDRSEL register holds an index to one of eight 24-bit pointers used to address program memory. The current program memory/external memory 24-bit address selected by the ADDRSEL register is accessible in the ADDRX (bits 23:16), ADDRH (bits 15:8), and ADDRL (bits 7:0) registers. The upper 5-bits of the ADDRSEL register are unused. All 8 banks of 24-bits are initialized to 0x000000 upon reset.

Program memory is always read or written as 16-bit words. On reads, the data from program memory is loaded into the DATAH and DATAL register pair. On writes, the contents of the DATAH and DATAL register pair are loaded into the program memory.

3.2 Data Memory

Figure 3-6 is a map of the data memory. The special-purpose registers and the first 128 data memory locations (between addresses 0x080 and 0x0FF) can be accessed with a direct addressing mode in which the absolute address of the operand is encoded within the instruction. The remaining 3840 bytes of data memory (between addresses 0x100 and 0xFFF) must be accessed using indirect or indirect-with-offset addressing modes. There is one 16-bit register for the indirect address pointer, and two 16-bit registers for indirect-with-offset address pointers. The offset is a 7-bit value encoded within the instruction. For more information about the addressing modes, see Section 4.1.

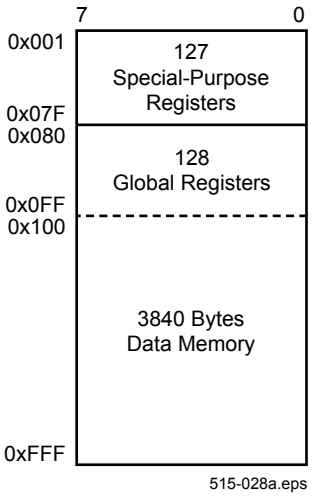


Figure 3-6 Data Memory Map

3.3 Program Memory

Figure 3-7 is a map of the program memory. A program memory address in the INTVECH/INVECL, IPCH/IPCL, or PCH/PCL registers or on the hardware stack is a word address. *However, the GNU software tools require byte addresses when referring to locations in program memory.* An address loaded in the ADDRX/ADDRH/ADDRL register is a byte address.

The program memory is organized as 8K-word pages (16K bytes). Single-instruction jumps and subroutine calls are restricted to be within the same page. Longer jumps and calls require using a **page** instruction to load the

upper address bits into the PA2:0 bits of the STATUS register. The **page** instruction must immediately precede the jump or call instruction. The PA2:0 bits should not be modified by writing directly to the STATUS register, because this may cause a mismatch between the PA2:0 bits in the STATUS register and the current program counter (see Section 3.3.2). For more information about the flash program memory, see Section 4.7 and Section 7.0.2.

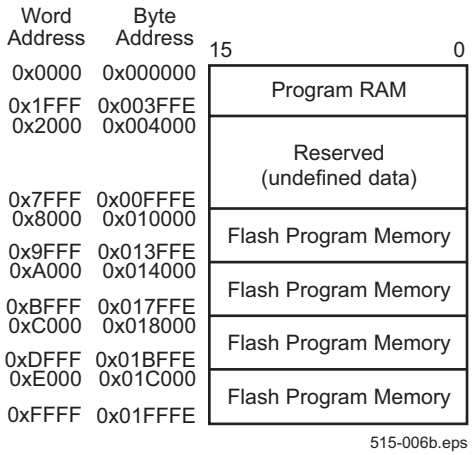


Figure 3-7 Program Memory Map

External memory is not shown in Figure 3-7 because the CPU cannot execute instructions directly out of external memory. For more information about external memory, see Section 5.11.

3.3.1 Loading the Program RAM

Software loads the program RAM from program flash memory using the **iread**/**ireadi** and **iwrite**/**iwritei** instructions. The **iread** instruction reads the 16-bit word specified by the address held in the ADDRX/ADDRH/ADDRL register. This word can be in program flash memory, program RAM, or external memory. When the **iread** instruction is executed, bits 15:8 of the word are loaded into the DATAH register, and bits 7:0 are loaded into the DATAL register. The address is a word-aligned byte address (i.e. an address that is zero in its LSB). The **ireadi** instruction is identical to the **iread** instruction, except that it also increments the address by 2.

The **iwrite** instruction writes the 16-bit word held in the DATAH/DATAL registers to the program RAM location specified by the address held in the ADDRX/ADDRH/ADDRL register. The **iwritei** instruction is identical, except that it also increments the address by 2. For more information about the **iread**/**ireadi** and **iwrite**/**iwritei** instructions, see Section 4.7.

3.3.2 Program Counter

The program counter holds the 16-bit address of the instruction to be executed. The lower eight bits of the program counter are held in the PCL register, and the upper eight bits are held in the PCH register. A write to the PCL register will cause a jump to the 16-bit address specified by the PCH and PCL registers. If the PCL register is written as the destination of an **add** or **addc** instruction and carry occurs, the PCH register is automatically incremented. (This may cause a mismatch between the PA2:0 bits in the STATUS register and the current program counter, therefore it is strongly recommended that direct modification of the PCL register is only used for jumps within a page.) The PCH register is read-only.

The PA2:0 bits in the STATUS register are not used for address generation, except when a jump or subroutine call instruction is executed. However, when an interrupt is taken, the PA2:0 bits are automatically updated with the upper three bits of the interrupt vector (INTVECH/L). These bits are restored from the STATUS shadow register when the interrupt service routine returns (i.e. executes a **reti** instruction).

3.4 Low Power Support

Software can change the execution speed of the CPU to reduce power consumption. A mechanism is also provided for automatically changing the speed on entry and return from the interrupt service routine. The **speed** instruction specifies power-saving modes that include a clock divisor between 1 and 128. This divisor only affects the clock to the CPU core, not the timers, SERDES, external memory or ADC (see Figure 3-16). The **speed** instruction also specifies the clock source (OSC clock, RTCLK oscillator, or PLL clock multiplier) and whether to disable the OSC clock oscillator or the PLL.

For maximum power savings when running from the OSC clock, disable the RTCLK oscillator (RTOSC_EN bit in the XCFG register), disable the watchdog timer (WDTE bit in the FUSE1 register), disable the A/D converter (ADCGO bit in the ADCCFG register, disable the analog comparator (CMPEN bit in the CMPCFG register) and check that no flash operation is in progress (FBUSY bit in the XCFG register) before executing a **speed #$FF** instruction.

To summarize settings for lowest power:

- XCFG bit 4 = 1

- FUSE1 bit 3 = 0
- CMPCFG bit 7 = 0
- ADCCFG bit 3 = 0
- XCFG bit 0 = 0

Note: Before executing the **speed** instruction or executing an interrupt (an interrupt will cause INTSPD to be copied to SPDREG), insure that the FCFG register has appropriate settings for the new clock frequency.

The SPDREG register (see Figure 3-4) holds the current settings for the clock divisor, clock source, and disable bits. These settings can be explicitly changed by executing a **speed** instruction, and they change automatically on interrupts. The SPDREG register is read-only, and its contents may only be changed by executing a **speed** instruction, taking an interrupt, or returning from an interrupt. Two consecutive **speed** instructions are not allowed. The INTSPD register specifies the settings used during execution of the interrupt service routine. The INTSPD register is both readable and writable.

On return from interrupts, the **reti** instruction includes a bit that specifies whether the pre-interrupt speed is restored or the current speed is maintained (see Table 3-5).

The actual speed of the CPU is indicated by the SPDREG register unless the specified speed is faster than the flash access time and the program is executing out of flash. When program execution moves from program RAM to program flash memory, the new clock divisor will be the greater (slower) of the clock divisor indicated by the SPDREG register and the clock divisor required to avoid violating the flash memory access time. The SPDREG register does not indicate if the flash clock divisor is being used. The speed indicated by the SPDREG will be overridden only if the speed is too fast for the flash memory.

The FCFG register holds bits that specify the minimum number of system clock cycles for each flash memory cycle (see Section 7.1.5).

3.4.1 Clock Stop Mode (SLEEP)

When a **speed** instruction occurs, it is possible for the CPU clock source to be disabled. The clock to the CPU core may be disabled while the system clock is left running, or the system clock may be disabled which also disables the CPU core clock. See SPDREG, Section 7.1.18.

3.4.2 Wakeup

Recovery from SLEEP (core clock stop) mode to normal execution is possible from these sources:

- External interrupts (i.e. Port B interrupts)
- Real-time timer interrupts
- Watchdog timer overflow reset
- Brown-out voltage reset
- $\overline{\text{RST}}$ external reset

The first two sources listed do not reset the chip, so register and CPU states are maintained. The last three sources reset the chip, so software must perform all of its reset initialization tasks to recover. This usually requires additional time, as compared to recovery through an interrupt. If a Port B or Real Time Timer interrupt occurs during core clock stop mode, the INTSPD register will be copied to the SPDREG register, the ISR will be executed, then mainline code will resume execution at the instruction after the **speed** command that caused the clock to stop.

Note: If wakeup triggers an ISR (Interrupt Service Routine) that has a **reti** instruction which reinstates the pre-interrupt speed (see Table 3-5), the device goes back to sleep. If a subsequent wakeup occurs which does not reinstate the pre-interrupt speed, then a **nop** must be inserted after the **speed** instruction which puts it to sleep.

3.5 Speed Change

The **speed** instruction executes using the current clock divisor. The new clock divisor takes effect with the following instruction, as shown in the following code example.

```
nop             ;assume divisor is 4, so this
                ;instruction takes 4 cycles
speed  #0x06    ;change the divisor to 8,
                ;instruction takes 4 cycles
nop             ;instruction takes 8 cycles
speed  #0x00    ;change the divisor to 1,
                ;instruction takes 8 cycles
nop             ;instruction takes 1 cycle
```

The automatic speed changes require a certain amount of delay to take effect (see Figure 3-4 and Figure 3-16):

- *Changing the Core Clock Divisor*—there is no delay when the clock divisor is changed (the instruction after the speed instruction is executed at the new speed).
- *Changing the System Clock Source*—the delay is up to one cycle of the slower clock. For example, changing between 4 MHz and 120 MHz could require up to 0.25 microseconds.

- *Turning on the OSC Clock Oscillator (clearing the $\overline{OSC}$ bit in the SPDREG register)*—the system clock suspend time is specified in the WUDX2:0 bits in the FUSE0 register (see Section 3.10.1).
- *Turning on the PLL Clock Multiplier (clearing the $\overline{PLL}$ bit in the SPDREG register)*—the system clock suspend time is specified in the WUDP2:0 bits in the FUSE0 register.

If both the OSC oscillator and PLL are re-enabled simultaneously, the delay is controlled by only the WUDP2:0 bits. Bits in the FUSE0 register are flash memory cells which cannot be changed dynamically during program execution.

3.6 Instruction Timing

All instructions that perform branches take 3 cycles to complete, consisting of 1 cycle to execute and 2 cycles to load the pipeline.

Table 3-3 Branch Timing

Instruction	Execution Time	Pipeline Load Time
`jmp`	1	2
`call`	1	2
`ret`	1	2
`reti`	1	2

In the case of an automatic speed change, the execution time will be with respect to the original speed and the pipeline load time will be with respect to the new speed.

Conditional branching is implemented in the IP2012 / IP2022 by using conditional skip instructions to branch over an unconditional jump instruction. To support conditional branching to other pages, the conditional skip instructions will skip over two instructions if the first instruction is a **`page`** instruction. The **`loadh`** and **`loadl`** instructions also cause an additional instruction to be skipped. When any of these conditions occur, it is called an *extended skip instruction*.

Skip instructions take 1 cycle if they do not skip, or 2 cycles if they skip over one instruction. An extended skip instruction may skip over more than one **`loadh`**, **`loadl`**, or **`page`** instruction, however this operation is interruptible and does not affect interrupt latency.

The **`iread`** and **`iwrite`** instructions take 4 cycles. The multiply instructions take 1 cycle.

3.7 Interrupt Support

There are three types of interrupt sources:

- *On-Chip Peripherals*–the serializer/deserializer units, real-time timer, timer 0, timer 1, and timer 2 are capable of generating interrupts. The Parallel Slave Peripheral does not generate interrupts on its own; it requires programming one of the Port B external interrupt inputs to generate interrupts on its behalf.
- *External Interrupts*–the eight pins on Port B can be programmed to generate interrupts on either rising or falling edges (see Section 5.1.1).
- ***`int`** Instruction*–the **`int`** instruction can be executed by software to generate an interrupt. The INT_EN bit can be considered as the interrupt flag for the **`int`** instruction, if the ISR (Interrupt Service Routine) checks for interrupt source. The INT_EN bit in the XCFG register must be set to enable the **`int`** instruction to trigger an interrupt. Because the **`reti`** instruction returns to the **`int`** instruction, the ISR must clear the INT_EN bit before returning.

Figure 3-8 shows the system interrupt logic. Each interrupt source has an interrupt enable bit. To be capable of generating an interrupt, the interrupt enable bit and the global interrupt enable (GIE) bit must be set.

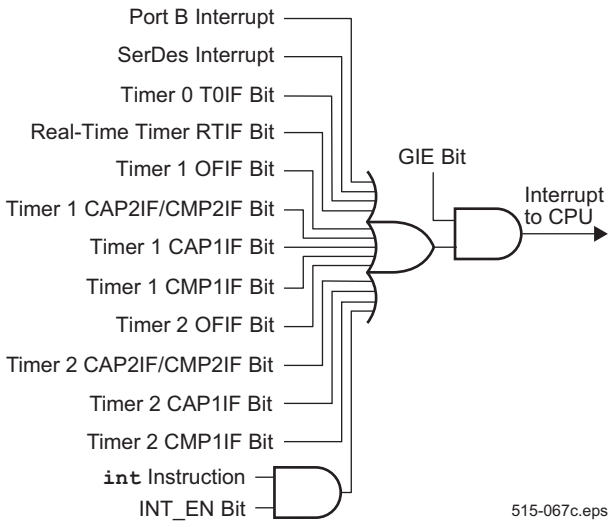


Figure 3-8 System Interrupt Logic

3.7.1 Interrupt Processing

There is one interrupt vector held in the INTVECH and INTVECL registers, which is reprogrammable by software. When an interrupt is taken, the current PC is saved in the IPCH and IPCL registers. On return from interrupt (i.e. execution of the **`reti`** instruction), the PC is restored from the IPCH and IPCL registers. Optionally, the

reti instruction may also copy the incremented PC to the INTVECH and INTVECL registers before returning. This has the effect of loading the INTVECH and INTVECL registers with the address of the next instruction following the **reti** instruction. This option can be used to directly implement a state machine, such as a simple round-robin scheduling mechanism for a series of interrupt service routines (ISRs) in consecutive memory locations.

If multiple sources of interrupts have been enabled, the ISR must check the interrupt flags of each source to determine the cause of the interrupt. The ISR must clear the interrupt flag for the source of the interrupt to prevent retriggering of the interrupt on completion of the ISR (i.e. execution of the **reti** instruction). Because the interrupt logic adds a 2-cycle delay between clearing an interrupt flag and deasserting the interrupt request to the CPU, the flag must be cleared at least 2 cycles before the **reti** instruction is taken.

When an interrupt is taken, the registers shown in Figure 3-9 are copied to a shadow register set. Each shadow register is actually a 2-level push-down stack, so one level of interrupt nesting is supported in hardware. The interrupt processing mechanism is completely independent of the 16-level call/return stack used for subroutines.

The contents of the DATAH and DATAL registers are pushed to their shadow registers 4 cycles after the interrupt occurs, to protect the result of any pending **iread** instruction. Therefore, software should not access the DATAH or DATAL registers during the first instruction of an ISR.

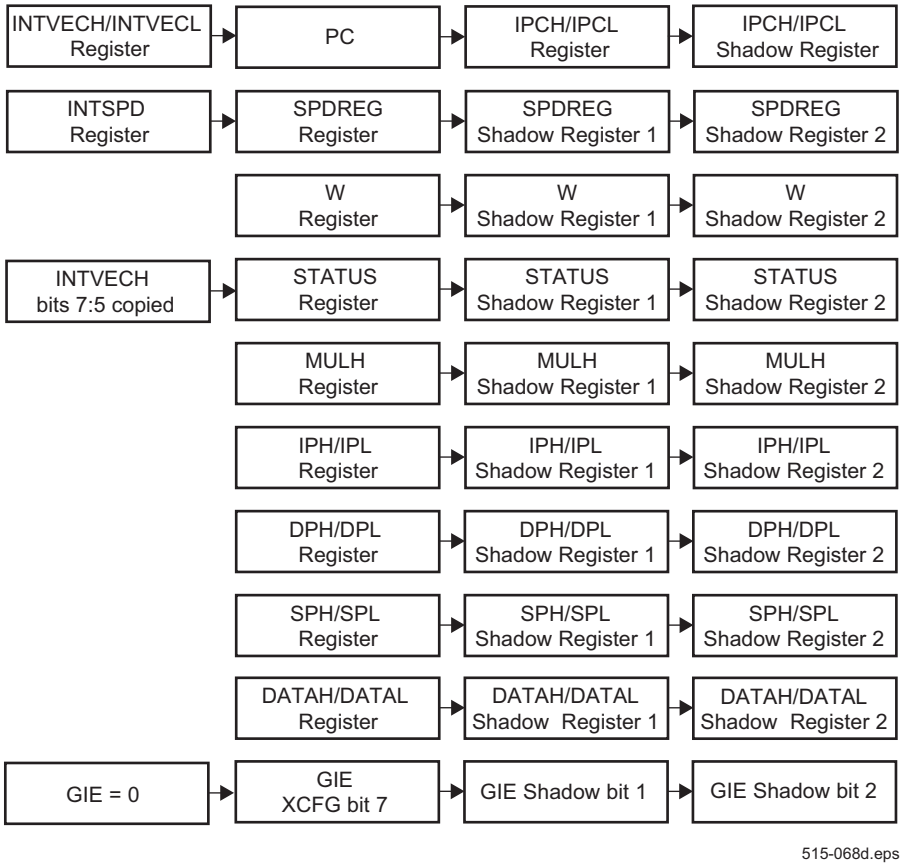


Figure 3-9 Interrupt Processing (On Entry to the ISR)

Note: On entry to the ISR the W, MULH, IPH/IPL, DPH/DPL, SPH/SPL, ADDRSEL and DATAH/DATAL register values don't change from their mainline code values (they are copied to their shadow registers).

On return from the ISR, these registers are restored from the shadow registers, as shown in Figure 3-10.

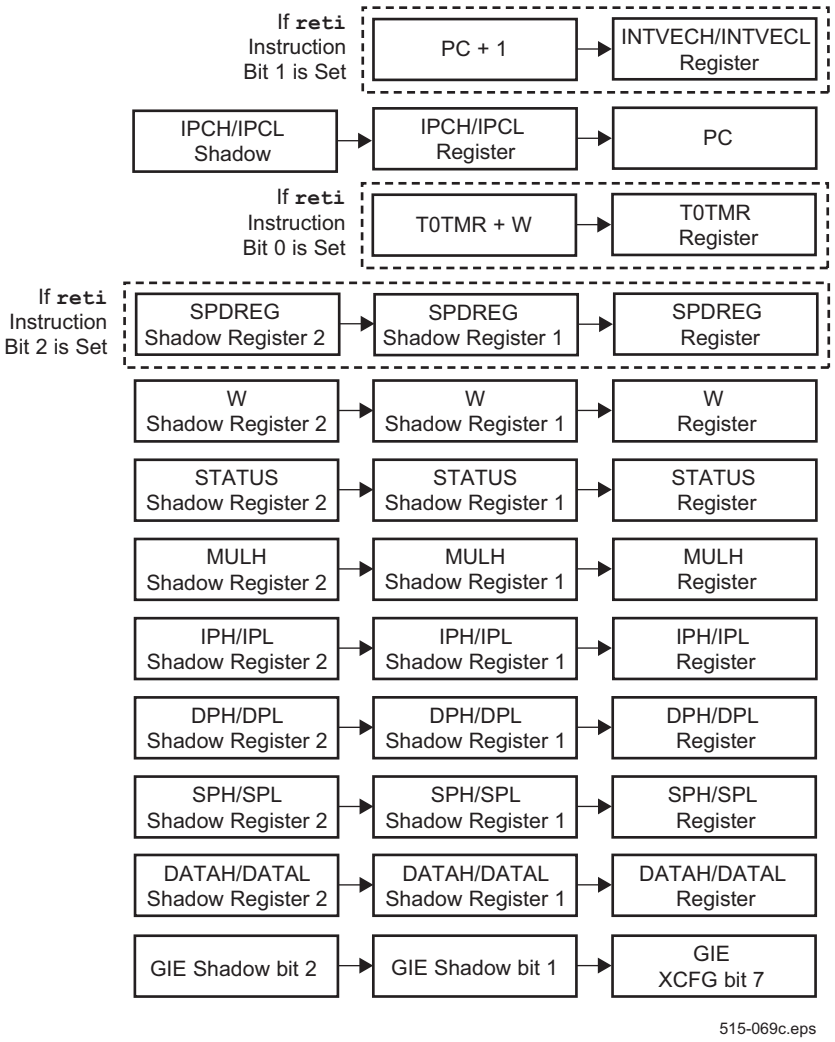


Figure 3-10 Interrupt Return Processing (upon execution of reti)

3.7.2 Global Interrupt Enable Bit

The GIE bit serves two purposes:

- Preventing an interrupt in a critical section of mainline code
- Supporting nested interrupts

The GIE bit is automatically cleared when an interrupt occurs, to disable interrupts while the ISR is executing. The GIE bit is automatically set by the **`reti`** instruction to re-enable interrupts when the ISR returns.

Table 3-4 GIE Bit Handling

Event	Effect
Enter ISR (interrupt)	GIE bit is cleared
Exit ISR (**`reti`** instruction)	GIE bit is set
`setb xcfg,7` instruction (inside ISR)	Enable interrupts for nested interrupt support
`clrb xcfg,7` instruction (inside ISR)	Nothing, the GIE bit is already clear
`setb xcfg,7` instruction (mainline code)	Enable interrupts
`clrb xcfg,7` instruction (mainline code)	Disable interrupts

To re-enable interrupts during ISR execution, the ISR code must first clear the source of the first interrupt. It may also be desirable to disable specific interrupts before setting the GIE bit to provide interrupt prioritization. Even with GIE deasserted, interrupt triggers are still captured but an interrupt won't be triggered until GIE is re-enabled. Caution must be taken not to exceed the interrupt shadow register stack depth of 2.

Clearing the GIE bit in the ISR cannot be used to globally disable interrupts so that they remain disabled when the ISR returns, because the **`reti`** instruction automatically sets the GIE bit. To disable interrupts in the ISR so that they remain disabled after the ISR returns, the individual interrupt enable bits for each source of interrupts must be cleared.

3.7.3 Interrupt Latency

The interrupt latency is the time from the interrupt event occurring to the first ISR instruction being latched from the decode to the execute stage (see Section 4.3). If the interrupt comes from a Port B input and the SYNC bit in the FUSE1 register is 0, an additional two core clock cycles of synchronization delay are added to the interrupt latency.

The **`ireadi`** or **`iwritei`** instructions are blocking (i.e. prevent other instructions and interrupts from being executed) for 4 core clock cycles. The **`iread`** or **`iwrite`** instructions are blocking for 4 core clock cycles while operating on program RAM, and non-blocking (single cycle) while operating on external memory.

When an interrupt event is triggered, the CPU speed is changed to the speed specified by the INTSPD register (the SPDREG register is copied to a shadow register, then loaded with the value from the INTSPD register).

If INTSPD is set the same as SPDREG when an interrupt occurs, then the interrupt latency is 3 core clock cycles for synchronous interrupts. If not, then the interrupt latency is 3 core clock cycles, plus the speed change (delay described in Section 3.5).

3.7.4 Return From Interrupt

The **`reti`** instruction word includes three bits which control its operation, as shown in Table 3-5. The three bits are specified from assembly language in a literal (e.g. **`reti #0x7`** to specify all bits as 1).

Table 3-5 `reti` Instruction Options

Bit	Function
2	Reinstate the pre-interrupt speed 1 = enable, 0 = disable
1	Store the PC+1 value in the INTVECH and INTVECL registers 1 = enable, 0 = disable
0	Add W to the T0TMR register 1 = enable, 0 = disable

Updating the interrupt vector allows the programmer to implement a sequential state machine. The next interrupt will resume the code directly after the previous **`reti`** instruction.

The **`reti`** instruction takes 1 cycle to execute, and there is a further delay of 2 cycles at the mainline code speed to load the pipeline before the mainline code is resumed.

Note: If RETI can return to Flash program memory, insure that all Flash reads or writes are complete (XCFG bit 0 = 0) before RETI is executed.

3.7.5 Disabled Interrupt Resources

If a peripheral is disabled and its interrupt flag is cleared, the peripheral does not have the ability to set an interrupt flag. The interrupt flag, however, is still a valid source of interrupt (If software sets an interrupt flag, the corresponding interrupt enable bit is set, and the GIE bit is set, then the CPU will be interrupted whether or not the peripheral is enabled or disabled).

If a peripheral is disabled inside the ISR, then its interrupt flag must be cleared to prevent an undesired interrupt from being taken when the ISR completes or when GIE is enabled (enabling nested interrupts - see Section 3.7.2).

3.8 Reset

There are five sources of reset:

- Power-On Reset (POR; reset occurs at power up)
- Brown-Out Reset (BOR)
- Watchdog Reset
- External Reset (from the $\overline{\text{RST}}$ pin)
- Tool Reset (from the debugging interface)

Each of these reset conditions causes the program counter to branch to the reset vector at the top of the program memory (word address 0xFFF0 or byte address 0x1FFE0).

The IP2012 / IP2022 incorporates a Power-On Reset (POR) detector that generates an internal reset as DVdd rises during power-up. Figure 3-11 is a block diagram of the reset logic. The startup timer controls the reset time-out delay. The reset latch controls the internal reset signal. On power-up, the reset latch is cleared (CPU held in reset), and the startup timer starts counting once it detects a valid logic high signal on the $\overline{\text{RST}}$ pin. Once the startup timer reaches the end of the timeout period, the reset latch is cleared, releasing the CPU from reset.

Note: CPU operation does not start until the CPU is released from reset and valid core clocks are received past the system clock suspend circuit (see WUDX block in Figure 3-16). So, for a POR with FUSE0 register WUDX=350us, for example, the core starts operation ~70ms (typically) after power up (35 ms min, 120 ms max). For a POR with WUDX= 1.1sec, the core starts operation ~1.1sec after power up.

The PSPCFG (address 0x06E) register contains two bits to indicate possible sources of the reset, WD and BO. The WD bit is cleared on reset unless the reset was caused by the watchdog timer, in which case the WD bit is set. The BO bit is cleared on reset unless the reset was caused by the brown-out logic, in which case, the BO bit is set.

Figure 3-12 shows a power-up sequence in which $\overline{\text{RST}}$ is not tied to the DVDD pin and the DVDD signal is allowed to rise and stabilize before $\overline{\text{RST}}$ pin is brought high. The WUDX2:0 bits of the FUSE0 register specify the length of time from the rising edge of $\overline{\text{RST}}$ until the device leaves reset.

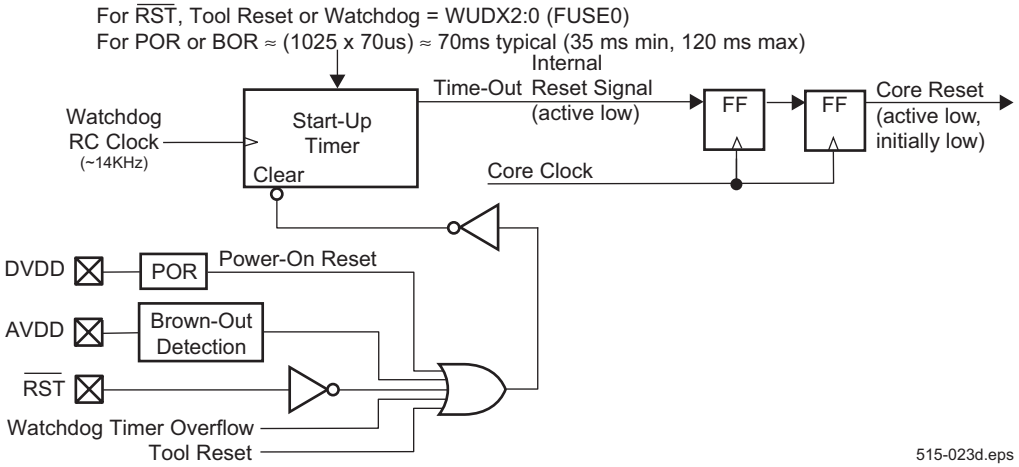


Figure 3-11 On-Chip Reset Circuit Block Diagram

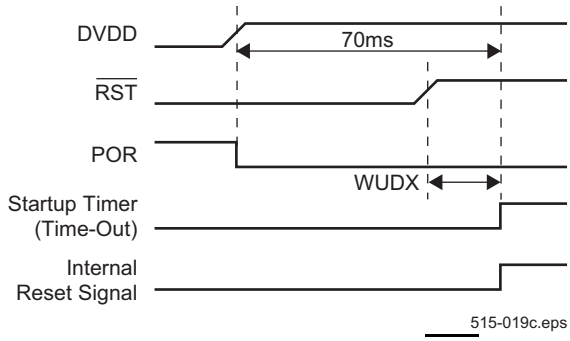


Figure 3-12 Power-Up, Separate RST Signal

Figure 3-13 shows the on-chip Power-On Reset sequence in which the RST and DVDD pins are tied together. The DVDD signal is stable before the startup timer expires. In this case, the CPU receives a reliable reset.

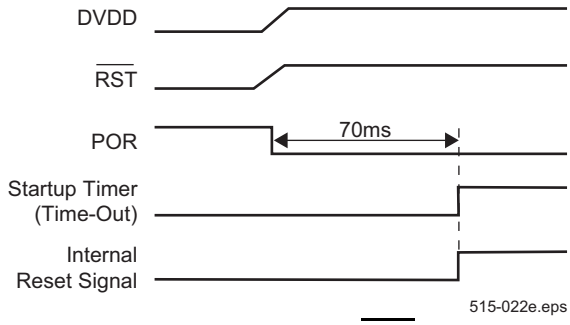


Figure 3-13 Power-On Reset, RST Tied To DVdd

However, Figure 3-14 depicts a situation in which DVDD rises too slowly. In this scenario, the startup timer will time out prior to DVDD reaching a valid operating voltage level (DVDD min). This means the CPU will come out of reset and start operating with the supply voltage below the level required for reliable performance. In this situation, an external RC circuit is recommended for driving RST. The RC delay should exceed five times the time period required for DVDD to reach a valid operating voltage.

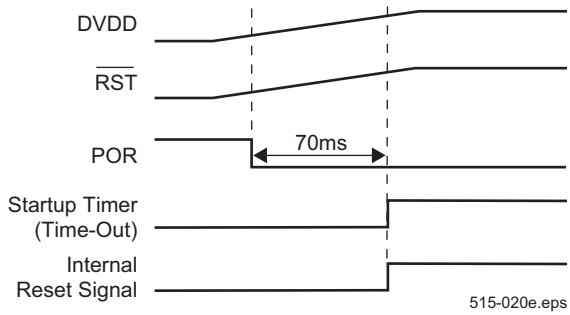


Figure 3-14 DVdd Rise Time Exceeds Tstartup

Figure 3-15 shows the recommended external reset circuit. The external reset circuit is required only if the DVDD rise time has the possibility of being too slow (refer to SVdd specification in Section 8.3).

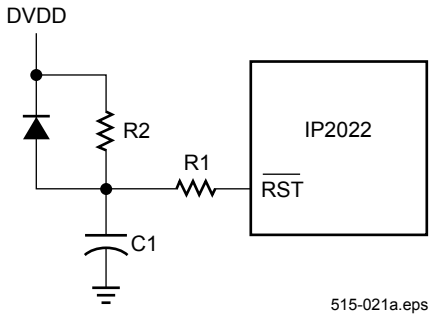


Figure 3-15 External Reset Circuit

The diode D discharges the capacitor when DVDD is powered down.

R1 = 100 Ω to 1K Ω will limit any current flowing into RST from external capacitor C1.

R2 < 40K Ω is recommended to make sure that voltage drop across R2 leaves the RST pin above a Vih level.

C1 should be chosen so that R2 x C1 exceeds five times the time period required for DVDD to reach a valid operating voltage.

R2 should not be tied to IOVdd, since IOVdd may reach the 3.3V level before DVdd is powered up.

3.8.1 Brown-Out Detector

The on-chip brown-out detection circuitry resets the CPU when AVdd dips below the brown-out voltage level programmed in the BOR2:0 bits of the FUSE1 register (refer to Section 3.10.2). Bits in the FUSE1 register are flash memory cells which cannot be changed dynamically during program execution.

The device is held in reset as long as AVdd stays below the brown-out voltage. The CPU will come out of reset when AVdd rises between 100mV and 300mV above the brown-out voltage setting (the CPU may never come out of brownout reset, even after AVdd returns to acceptable level, if the brownout setting is too high). Therefore, the 2.10V setting is recommended. The brown-out level can be programmed using the BOR2:0 bits in the FUSE1 register, as shown in Section 3.10.2. Brown-Out should be enabled, so that the Flash memory does not get corrupted when powering down.

3.8.2 Reset and Interrupt Vectors

After reset, the PC is loaded with 0xFFF0, which is near the top of the program memory space. Typical activities for the reset initialization code include:

- Setting up the FCFG register with appropriate values for flash timing compensation.
- Issuing a **`speed`** instruction to initialize the CPU core clock speed.
- Checking for the cause of reset (brown-out voltage, watchdog timer overflow, or other cause). In some applications, a “warm” reset allows some data initialization procedures to be skipped.
- Copying speed-critical sections of code from flash memory to program RAM.
- Setting up data memory structures (stacks, tables, etc.).
- Initializing peripherals for operation (timers, etc.).
- Initializing the dynamic interrupt vector and enabling interrupts.

Because the default interrupt vector location is 0, which is in program RAM, interrupts should not be enabled until the ISR is loaded in shadow RAM or the dynamic interrupt vector is loaded with the address of an ISR in flash memory. There is a single dynamic interrupt vector shared by all interrupts. The interrupt vector can be changed by loading the INTVECH and INTVECL registers, or by issuing a **`reti`** instruction with an option specifying that the interrupt vector should be updated with the current PC value plus 1.

3.8.3 Register States Following Reset

The effect of different reset sources on a register depends on the register and the type of reset operation. Some registers are initialized to specific values, some are left unchanged, and some are undefined.

A register that starts with an unknown value should be initialized by the software to a known value if it is going to be used (no need to initialize unused registers nor data memory). Do not simply test the initial state and rely on it starting in that state consistently. See Table 7-1 for more detailed information.

3.9 Clock Circuitry

There are two clock oscillators, the OSC oscillator and the RTCLK oscillator. Using the PLL clock multiplier, the OSC clock is intended to provide the time base for running the CPU core at speeds up to 120MHz. The RTCLK oscillator operates at 32.768kHz using an external crystal. This oscillator is intended for running the real-time timer when the OSC oscillator and PLL clock multiplier are turned off. Either clock source can be driven by an external clock signal up to 120MHz.

Figure 3-16 shows the clock logic. The PLL clock multiplier has a fixed multiplication factor of 50. The PLL is preceded by a divider capable of any integer divisor between 1 and 8, as controlled by the PIN2:0 bits of the FUSE0 register (refer to Section 3.10.1). The PLL is followed by a second divider capable of any integer divisor between 1 and 4, as controlled by the POUT1:0 bits of the FUSE0 register. A third divider which only affects the clock to the CPU core is controlled by the speed change mechanism described in Section 3.4. See Section 3.10.1 for a description of the FUSE0 WUDX2:0 and WUDP2:0 bits.

Note: Bits in the FUSE0 register are flash memory cells which cannot be changed dynamically during program execution.

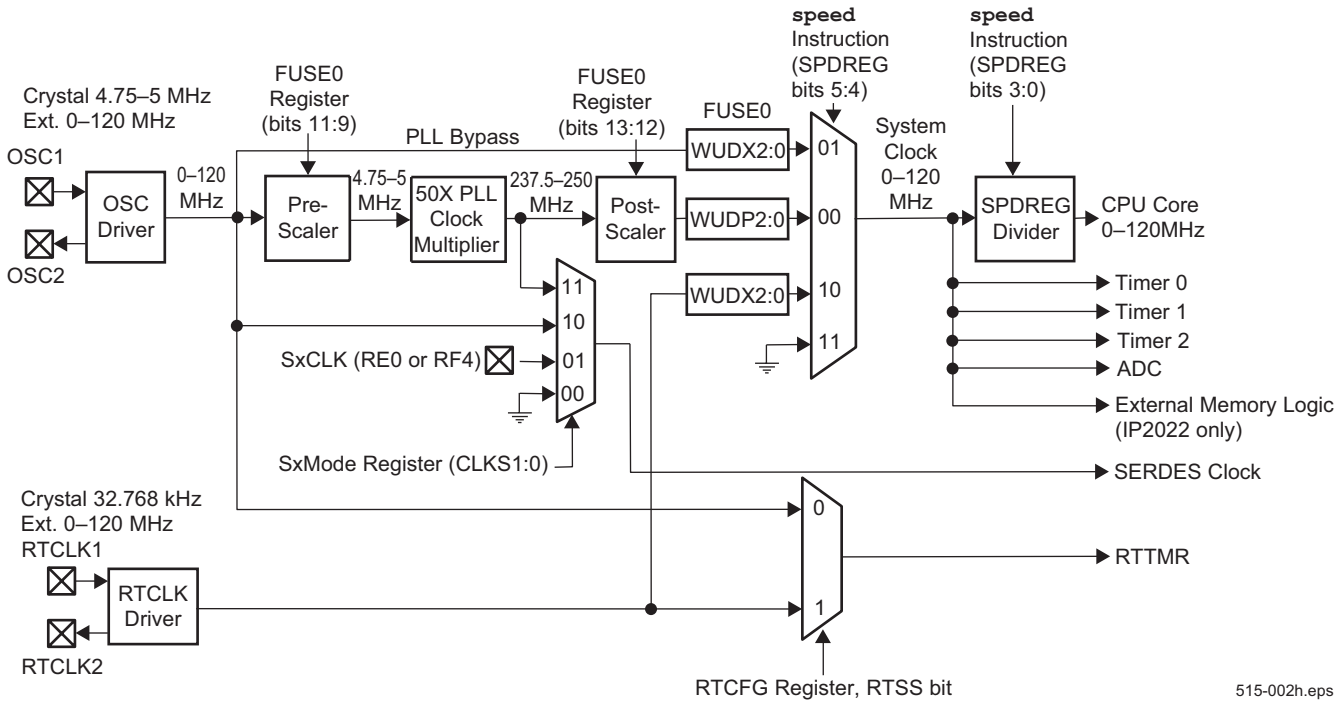


Figure 3-16 Clock Logic

3.9.1 External Clock / Crystal Connections

Figure 3-17 shows the connections for driving the OSC and/or RTCLK clock sources with an external signal. To drive the OSC clock source, the external clock signal is driven on the OSC1 pin and the OSC2 pin is left open. The external clock signal driven on the OSC1 pin may be any frequency up to 120 MHz. To drive the RTCLK clock source, the external clock signal is driven on the RTCLK1 input and the RTCLK2 output is left open. The external clock signal driven on the RTCLK1 pin may be any frequency up to 120 MHz.

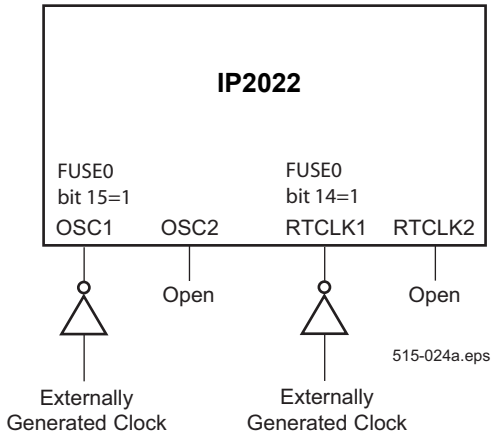


Figure 3-17 External Clock Inputs

Figure 3-18 shows the connections for attaching a crystal to the OSC and/or RTCLK oscillator. For the OSC oscillator, a 4.8MHz crystal is connected across the OSC1 and OSC2 pins. For the RTCLK oscillator, a 32.768kHz crystal is connected across the RTCLK1 and RTCLK2 pins.

There is about 4pf of capacitance on each of OSC1 and OSC2 pins to DVss and about 10pf of capacitance on each of RTCLK1 and RTCLK2 pins to DVss. There is also an internal feedback resistor (no external feedback resistor needed). For the OSC crystal, a parallel resonant crystal is recommended that has a maximum ESR of 50 ohms. For the optional RTCLK crystal, a parallel resonant crystal is recommended that has a maximum ESR of 50K ohms.

The crystal manufacturer's load capacitance rating (C_L) should be equal to (C_1 x C_2)/(C_1 + C_2), where C_1 = capacitance on OSC1 (4pF + stray board capacitance + added capacitance), and C_2 = capacitance on OSC2 (4pF + stray board capacitance + added capacitance). The trace length between the OSC pins and the crystal should be as short as possible, to avoid noise coupling.

When RTCLK1 is unused, it should be tied to GND, with RTCLK2 open.

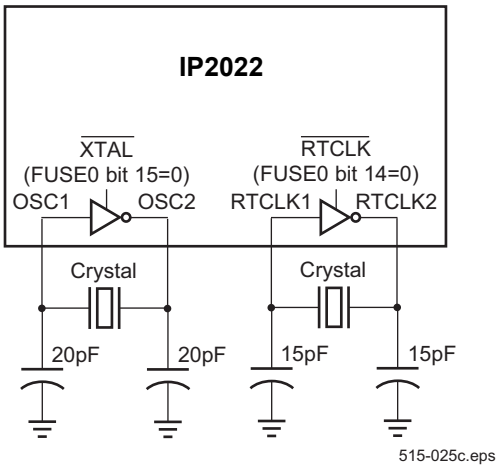


Figure 3-18 Crystal Connection

3.10 Configuration Block

The configuration block is a set of flash memory registers outside of both program memory and data memory. These registers are not readable or writable at run time.

The FUSE0, FUSE1 registers hold settings that must be specified by system designers. The other configuration block registers are used by software tools. The configuration block is readable but not writable when Code Protection is enabled. Table 3-6 lists the configuration block registers.

Table 3-6 Configuration Block

Word Address	Words	Name	Description
0x00010000	1	FUSE0	FUSE0 register
0x00010001	1	FUSE1	FUSE1 register
0x00010002 to 0x00010003	2	-	Reserved
0x00010004	1	TRIM0	TRIM0 register, factory programmed to FBFE
0x00010005 to 0x0001001D	25	-	Reserved
0x0001001E- 0x0001001F	2	FREQ	OSC1 input frequency during device programming - used by tools only
0x00010020 to 0x00010027	8	VCOMPANY	Company name
0x00010028 to 0x0001002F	8	VPRODUCT	Product name
0x00010030 to 0x00010031	2	VVERSION	Software version
0x00010032 to 0x00010033	2	VSOFTDATE	Software date
0x00010034 to 0x00010035	2	VPROGDATE	Programming date
0x00010036 to 0x0001003F	10	-	Reserved
Total 16-bit words	64		

3.10.1 FUSE0 Register (not run-time programmable)

15	14	13 12	11 10 9	8	7 6	5 4	3 2 1 0
$\overline{\text{XTAL}}$	$\overline{\text{RTCLK}}$	POUT1:0	PIN2:0	Reserved	WUDP2:0	WUDX2:0	

Figure 3-19 FUSE0 Register

$\overline{\text{XTAL}}$ OSC2 crystal drive output
- 0 = Enabled — Use for crystal clock
- 1 = Disabled — Use for external clock

$\overline{\text{RTCLK}}$ RTCLK2 crystal drive output
- 0 = Enabled — Use for crystal clock
- 1 = Disabled — Use for external clock

POUT1:0 Specifies PLL clock multiplier postscaler divisor
- 00 = 1 (reserved)
- 01 = 2
- 10 = 3
- 11 = 4

PIN2:0 Specifies PLL clock multiplier prescaler divisor
- 000 = 1
- 001 = 2
- 010 = 3
- 011 = 4
- 100 = 5
- 101 = 6
- 110 = 7
- 111 = 8

WUDP2:0 Specifies system clock suspend time during PLL startup (after a **speed** instruction clears the $\overline{\text{PLL}}$ bit in the SPDREG register). Times listed here are typical (2.5V, 22°C). Use -50% and +70% for worst case.
- 000 = 140 µs
- 001 = 210 µs
- 010 = 350 µs
- 011 = 630 µs
- 100 = 1 ms
- 101 = 2 ms
- 110 = 5 ms
- 111 = 9 ms

WUDX2:0 Specifies system clock suspend time during OSC and RTCLK start-up. Used to keep the clock from propagating to the core before the crystal achieves valid signal levels (see Figure 3-16). Also keeps $\overline{\text{RST}}$ asserted except for POR and BOR. Times listed here are typical (2.5V, 22°C). Use -50% and +70% for worst case.
- 000 = 450 µs
- 001 = 1 ms
- 010 = 5 ms
- 011 = 9 ms
- 100 = 18 ms
- 101 = 72 ms
- 110 = 574 ms†
- 111 = 1147 ms†

†. Clock suspend time after POR is twice this long if the Watchdog is enabled in FUSE1 to a value less than WUDX.

3.10.2 FUSE1 Register (not run-time programmable)

15	14	13 … 7	6 5 4	3	2 1 0
$\overline{CP}$	$\overline{SYNC}$	Reserved	BOR2:0	WDTE	WDPS2:0

Figure 3-20 FUSE1 Register

$\overline{CP}$

Clear to enable code protection. Once cleared, this bit cannot be set until the entire device is erased. When code protection is enabled, program memory reads as all 0s to an external device programmer, and writes to the program memory will be ignored. This bit does not affect access to program memory made by software, using the **`iread, ireadi, iwrite, iwritei, ferase, fwrite`** and **`fread`** instructions. In-system debugging is not available when code protection is enabled. Code protection does not protect the configuration block against reading, only against writing. Note: After clearing this bit during programming, Code Protect is not activated until the part is powered down or reset.

0 = enabled
1 = disabled

$\overline{SYNC}$

Set to read directly from the port pins through the RxIN register, clear to read through a CPU core clock synchronization register. This bit should be clear if any external devices that can be read from I/O port pins are running asynchronously to the CPU core clock. See Figure 5-1.

0 = enabled
1 = disabled

BOR2:0

Specifies brown-out voltage level. If AVdd goes below this level, the IP2012 / IP2022 is reset. There is a maximum of 0.3V hysteresis to leave brownout reset after brownout reset occurs. Brownout hysteresis does not apply to power-up.

000 = 2.30V ± 0.1V Do not use unless AVdd ≥ 2.60V
001 = 2.25V ± 0.1V Do not use unless AVdd ≥ 2.55V
010 = 2.20V ± 0.1V Do not use unless AVdd ≥ 2.50V
011 = 2.15V ± 0.1V
100 = 2.10V ± 0.1V Should use this setting.
101 = Reserved
110 = Reserved
111 = Disabled, no brown-out reset can occur. WARNING - Flash memory corruption may occur during power-down if this is selected.

WDTE

Enables Watchdog Timer in run mode. Disabled in debug mode regardless of this bit.

0 = disabled
1 = enabled

WDPS2:0

Specifies the Watchdog Timer time period before the Watchdog Timer expires. If the Watchdog Timer is enabled, software must clear the Watchdog Timer periodically within this time period to prevent a reset of the IP2012 / IP2022 from occurring. The **`cwdt`** instruction or any reset clears both the Watchdog Timer and its prescaler. Care must be taken to ensure that this setting is greater than the maximum crystal startup time plus the time required to get to the first **`cwdt`** instruction. Times listed here are typical (2.5V, 22°C). Use -50% and +70% for worst case.

000 = 20 ms
001 = 40 ms
010 = 80 ms
011 = 160 ms
100 = 320 ms
101 = 640 ms
110 = 1280 ms
111 = 2560 ms

3.10.3 TRIM0 Register

Factory programmed to $FBFE. Must leave at FBFE for 4.75 - 5 MHz into the PLL.

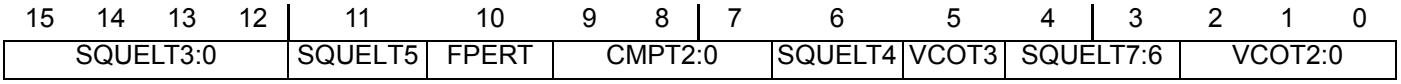


Figure 3-21 TRIM0 Register

SQUELT7:0	SERDES squelch trim bits
FPERT	Controls flash block pulse erase, both for self-programming **`ferase`** and for the FERASE command from the ISD/ISP interface 0 = 20 ms, if OSC1 frequency and FCFG register settings are optimal 1 = Reserved - 10ms block erase (do not use)
CMPT2:0	Comparator offset trim bits
VCOT3:0	PLL VCO frequency trim bits 1110 = 4.75–5.0 MHz into PLL 1101 = 3.2 MHz into PLL

4.0 Instruction Set Architecture

The IP2012 / IP2022 implements a powerful load-store RISC architecture with a rich set of arithmetic and logical operations, including signed and unsigned 8-bit x 8-bit integer multiply with a 16-bit product.

The CPU operates on data held in 128 special-purpose registers, 128 global registers, and 3840 bytes of data memory. The special-purpose registers are dedicated to control and status functions for the CPU and peripherals. The global registers and data memory may be used for any functions required by software, the only distinction among them being that the 128 global registers (addresses 0x080 to 0x0FF) can be accessed using a direct addressing mode. The remaining 3840 bytes of data memory (between addresses 0x100 and 0xFFF) must be accessed using indirect or indirect-with-offset addressing modes. The IPH/IPL register is the pointer for the indirect addressing mode, and the DPH/DPL and SPH/SPL registers are the pointers for the indirect-with-offset addressing modes.

4.1 Addressing Modes

A 9-bit field within the instruction, called the “fr” field, specifies the addressing mode and the address (in the case of direct addressing) or the address offset (in the case of indirect-with-offset addressing), as shown in Table 4-1. (See Figure 3-6 for data RAM map.)

4.1.1 Pointer Registers

When an addition or increment instruction (i.e. **`add`**, **`inc`**, **`incsz`**, or **`incsnz`**) on the low byte of a pointer register (i.e. IPL, DPL, SPL, or ADDRL) generates a carry, the high part of the register is incremented. For example, if the IP register holds 0x00FF and an **`inc ipl`** instruction is executed, the register will hold 0x0100 after the instruction. When a subtraction or decrement instruction (i.e. **`sub`**, **`subc`**, **`dec`**, **`decsz`**, or **`decsnz`**) generates a borrow, the high part of the register is decremented.

Note: Because carry and borrow are automatically handled, the **`addc`** and **`subc`** instructions are not needed for arithmetic operations on pointer registers.

Table 4-1 Addressing Mode Summary

“fr” Field	Mode	Syntax	Effective Address (EA)	Restrictions
0 0000 0000	Indirect	`mov w,(ip)` `mov (ip),w`	IPH || IPL	0x020 ≤ EA ≤ 0xFFF
0 0nnn nnnn	Direct, special-purpose registers	`mov w,fr` `mov fr,w`	nnnnnnn	0x002 ≤ EA ≤ 0x07F
0 1nnn nnnn	Direct, global registers	`mov w,fr` `mov fr,w`	0x080 + nnnnnnn	0x080 ≤ EA ≤ 0x0FF
1 0nnn nnnn	Indirect with offset, data pointer	`mov w,offset(dp)` `mov offset(dp),w`	DPH || DPL + nnnnnnn	0x000 ≤ nnnnnnn ≤ 0x07F 0x020 ≤ EA ≤ 0xFFF
1 1nnn nnnn	Indirect with offset, stack pointer	`mov w,offset(sp)` `mov offset(sp),w`	SPH || SPL + nnnnnnn	0x000 ≤ nnnnnnn ≤ 0x07F 0x020 ≤ EA ≤ 0xFFF

4.1.2 Direct Addressing Mode

Figure 4-1 shows the direct addressing mode used to reference the special-purpose registers. Seven bits from the “fr” field allow addressing up to 128 special-purpose registers. (Not all 128 locations in this space are implemented in the IP2012 / IP2022; several locations are reserved for future expansion.)

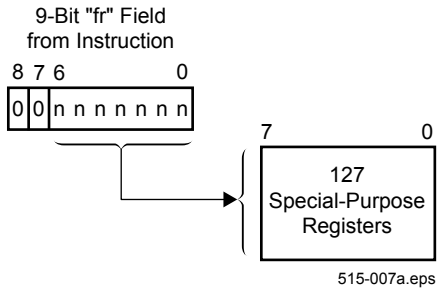


Figure 4-1 Direct Mode, Special-Purpose Registers

The following code example uses direct mode.

```
mov    w,0x0012   ;load W with the contents of
                  ;the memory location at 0x0012
                  ;(the DATAL register)
```

Figure 4-2 shows the direct addressing mode used to reference the global registers. This mode is distinguished from the mode used to access the special-purpose registers with bit 7 of the “fr” field. Because these registers have this additional addressing mode not available for the other data memory locations, they are especially useful for holding global variables and frequently accessed data.

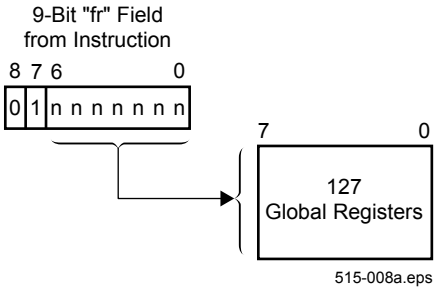


Figure 4-2 Direct Mode, Global Registers

Note: Addresses from 0x000 to 0x01F can only be accessed with Direct mode.

4.1.3 Indirect Addressing Mode

The indirect addressing mode is used when all of the bits in the “fr” field are clear. The location of the operand is specified by a 12-bit pointer in the IPH and IPL registers. The upper four bits of the IPH register are not used. Figure 4-3 shows indirect mode.

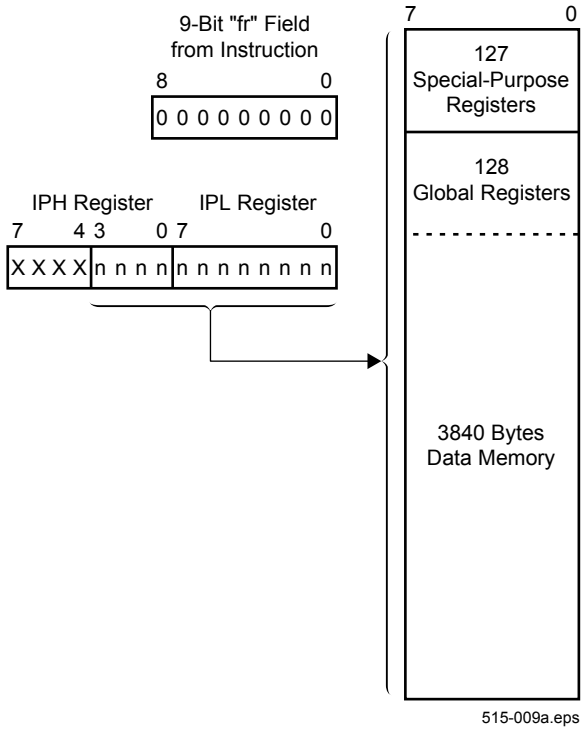


Figure 4-3 Indirect Mode

The following code example uses indirect mode.

```
mov    w,#0x03    ;load W with 0x03
mov    iph,w      ;load the high byte of the
                  ;indirect pointer from W
mov    w,#0x85    ;load W with 0x85
mov    ipl,w      ;load the low byte of the
                  ;indirect pointer from W
mov    w,(ip)     ;load W with the contents of
                  ;the memory location at
                  ;effective address 0x0385
```

4.1.4 Indirect-with-Offset Addressing Mode

The indirect-with-offset addressing mode is used when bit 8 of the “fr” field is set. The location of the operand is specified by a 7-bit unsigned immediate from the “fr” field added to a 12-bit base address in a pointer register.

When bit 7 of the “fr” field is clear, the DPH/DPL register is selected as the pointer register. This register is accessed using the **`loadh`** and **`loadl`** instructions, which load its high and low bytes, respectively. The upper four bits of the DPH register are not used. Figure 4-4 shows indirect-with-offset addressing using the DPH/DPL register as the pointer register.

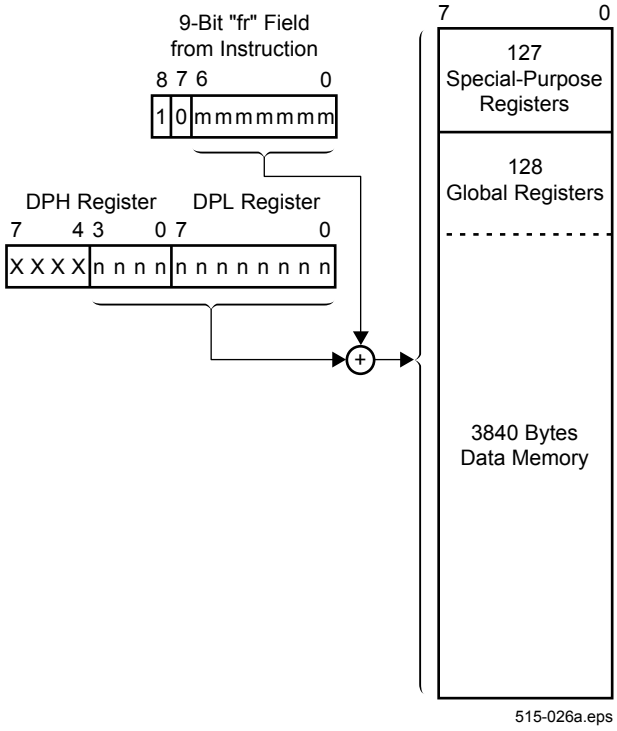


Figure 4-4 Indirect-with-Offset Mode, Data Pointer

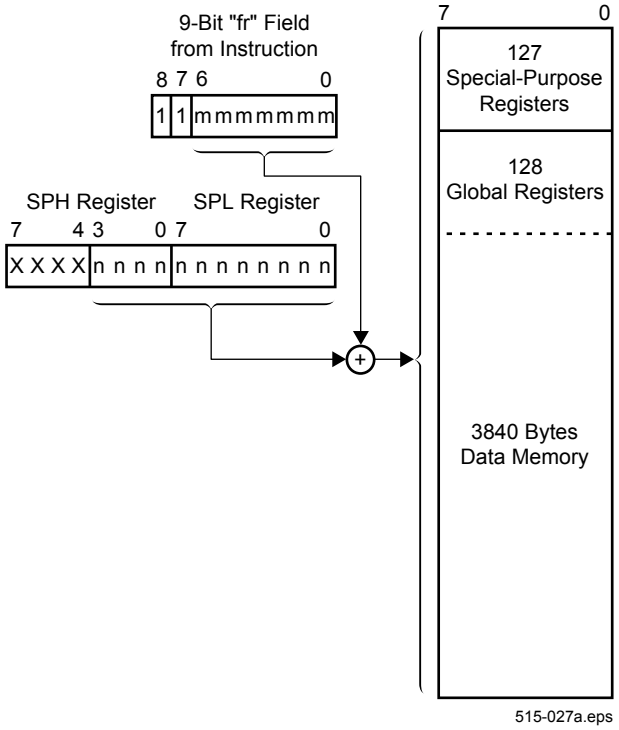
The following code example uses indirect-with-offset addressing mode.

```
mov       w,#$03    ;load W with $03
mov       dph,w     ;load the high byte of the
                    ;DPH/DPL register from W
mov       w,#$85    ;load W with $85
mov       dpl,w     ;load the low byte of the
                    ;DPH/DPL register from W
mov       w,8(dp)   ;load W with the contents of
                    ;the memory location at
                    ;address 0x038D
                    ;(i.e. 0x0385 + 0x0008)
```

When bit 7 of the “fr” field is set, the SPH/SPL register is selected as the pointer register. The upper four bits of the SPH register are not used. Figure 4-5 shows indirect-with-offset mode using the SPH/SPL register. In addition to this indirect-with-offset addressing mode, there are also **`push`** and **`pop`** instructions which automatically increment and decrement the SPH/SPL register while performing a data transfer between the top of stack and a data memory location specified by the “fr” field. Stacks grow down from higher addresses to lower addresses. This stack addressing mechanism is completely independent from the hardware stack used for subroutine call and return.

When a **`pop`** instruction is used with the indirect-with-offset addressing mode, the address calculation for the “fr” operand is made using the value in the SPH/SPL register *before* the automatic increment, even though the stack operand itself is addressed using the value *after* the automatic increment.

Figure 4-5 Indirect-with-Offset Mode, Stack Pointer



4.2 Instruction Set

The instruction set consists entirely of single-word (16-bit) instructions, most of which can be executed at a rate of one instruction per clock cycle, for a throughput of up to 120 MIPS when executing out of program RAM.

Assemblers may implement additional instruction mnemonics for the convenience of programmers, such as a long jump instruction which compiles to multiple IP2012 / IP2022 instructions for handling the page structure of program memory. Refer to the assembler documentation for more information about any instruction mnemonics implemented in the assembler.

4.2.1 Instruction Formats

There are five instruction formats:

- Two-operand arithmetic and logical instructions
- Immediate-operand arithmetic and logical instructions
- Jumps and subroutine calls
- Bit operations
- Miscellaneous instructions

Figure 4-6 shows the two-operand instruction format. The two-operand instructions perform an arithmetic or logical operation between the W register and a data memory location specified by the “fr” field. The D bit indicates the destination operand. When the D bit is clear, the destination operand is the W register. When the D bit is set, the destination operand is specified by the “fr” field.

There are some exceptions to this behavior. The multiply instructions always load the 16-bit product into the MULH and W registers. The MULH register receives the upper 8 bits, and the W register receives the lower 8 bits.

Traditionally single-operand instructions, such as increment, are available in two forms distinguished by the D bit. When the D bit is clear, the source operand is specified by the “fr” field and the destination operand is the W register. When the D bit is set, the data memory location specified by the “fr” field is both the source and destination operand.

Also, there are a few cases of unrelated instructions, such as **`clr`** and **`cmp`**, which are distinguished by the D bit.

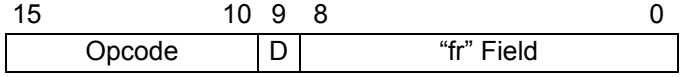


Figure 4-6 Two-Operand Instruction Format

Figure 4-7 shows the immediate operand instruction format. In this format, an 8-bit literal value is encoded in the instruction field. Usually the W register is the destination operand, however this format also includes instructions that use the top of the stack or a special-purpose register as the destination operand.

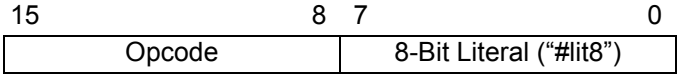


Figure 4-7 Immediate-Operand Instruction Format

Figure 4-8 shows the format of the jump and subroutine call instructions. 13 bits of the entry point address are encoded in the instruction. The remaining three bits come from the PA2:0 bits of the STATUS register.

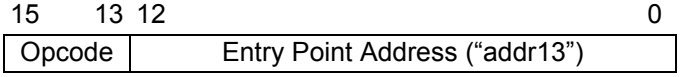


Figure 4-8 Jump and Call Instruction Format

Figure 4-9 shows the format of the instructions that clear, set, and test individual bits within registers. The register is specified by the “fr” field, and a 3-bit field in the instruction selects one of the eight bits in the register.

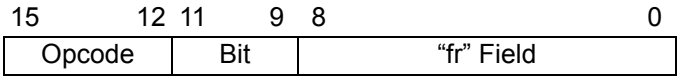


Figure 4-9 Bit Operation Instruction Format

Figure 4-10 shows the format of the remaining instructions.

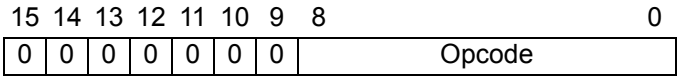


Figure 4-10 Miscellaneous Instruction Format

4.2.2 Instruction Types

The instructions are grouped into the following functional categories:

- Logical instructions
- Arithmetic and shift instructions
- Bit operation instructions
- Data movement instructions
- Program control instructions
- System control instructions

Logical Instructions

Each logic instruction performs a standard logical operation (AND, OR, exclusive OR, or logical complement) on the respective bits of the 8-bit operands. The result of the logic operation is written to W or to the data memory location specified by the “fr” field.

All of these instructions take one clock cycle for execution.

Arithmetic and Shift Instructions

Each arithmetic or shift instruction performs an operation such as add, subtract, add with carry, subtract with carry,

rotate left or right through carry, increment, decrement, clear to zero, or swap high/low nibbles. The compare (**cmp**) instruction performs the same operation as subtract, but it only updates the C, DC, and Z flags of the STATUS register; the result of the subtraction is discarded.

There are instructions available (**incsz**, **decsz**) that increment or decrement a register and simultaneously test the result. If the 8-bit result is zero, the next instruction in the program is skipped. These instructions can be used to make program loops. There are also compare-and-skip instructions (**cse**, **csne**) which perform the same operation as subtract, but perform a conditional skip without affecting either operand or the condition flags in the STATUS register.

All of the arithmetic and shift instructions take one clock cycle for execution, except in the case of the test-and-skip instructions when the tested condition is true and a skip occurs, in which case the instruction takes at least two cycles. If a skip instruction is immediately followed by a **loadh**, **loadl**, or **page** instruction (and the tested condition is true) then two instructions are skipped and the operation consumes three cycles. This is useful for skipping over a conditional branch to another page, in which a **page** instruction precedes a **jmp** instruction. If several **page** or **loadh**/**loadl** instructions immediately follow a skip instruction, then they are all skipped plus the next instruction and a cycle is consumed for each. These "extended skip instructions" are interruptible, so they do not affect interrupt latency.

Bit Operation Instructions

There are four bit operation instructions:

- **setb**—sets a single bit in a data register without affecting other bits
- **clrb**—clears a single bit in a data register without affecting other bits
- **sb**—tests a single bit in a data register and skips the next instruction if the bit is set
- **snb**—tests a single bit in a data register and skips the next instruction if the bit is clear

All of the bit operation instructions take one clock cycle for execution, except for test-and-skip instructions when the tested condition is true and a skip occurs.

Data Movement Instructions

A data movement instruction moves a byte of data from a data memory location to either the W register or the top of stack, or it moves the byte from either the W register or the top of stack to a data memory location. The location is specified by the "fr" field. The SPH/SPL register pair points to the top of stack. This stack is independent of the hardware stack used for subroutine call and return.

Program Control Instructions

A program control instruction alters the flow of the program by changing the contents of the program counter. Included in this category are the jump, call, return-from-subroutine, and interrupt instructions.

The **jmp** instruction has a single operand that specifies the entry point at which to continue execution. The entry point is typically specified in assembly language with a label, as in the following code example:

```
snb     status,0   ;test the carry bit
jmp     do_carry   ;jump to do_carry routine
                   ;if C = 1


...
do_carry:          ;jump destination label
...                ;execution continues here
```

If the carry bit is set to 1, the **jmp** instruction is executed and program execution continues where the **do_carry** label appears in the program.

The **call** instruction works in a similar manner, except that it saves the contents of the program counter to the CALLH/CALLL registers before jumping to the new address. It calls a subroutine that is terminated by a **ret**, **retw**, or **retnp** instruction, as shown in the following code example:

```
call    add_2bytes  ;call subroutine
                    ;add_2bytes
nop                 ;execution returns to
                    ;here after the
                    ;subroutine is finished
...
add_2bytes:         ;subroutine label
...                 ;subroutine code goes
                    ;here
ret                 ;return from subroutine
```

Returning from a subroutine restores the saved program counter contents from the CALLH/CALLL registers, which causes program to resume execution with the instruction immediately following the **call** instruction (a **nop** instruction, in the above example)

A program memory address contains 16 bits. The **jmp** and **call** instructions specify only the lowest thirteen bits of the jump/call address. The upper 3 bits come from the PA2:0 bits of the STATUS register. An indirect relative jump can be accomplished by adding the contents of the W register to the PCL register (i.e. an **add pcl,w** instruction).

Program control instructions such as **jmp**, **call**, and **ret** alter the normal program sequence. When one of

these instructions is executed, the execution pipeline is automatically cleared of pending instructions and refilled with new instructions, starting at the new program address. Because the pipeline must be cleared, three clock cycles are required for execution, one to execute the instruction and two to reload the pipeline.

System Control Instructions

A system control instruction performs a special-purpose operation that sets the operating mode of the device or reads data from the program memory. Included in this category are the following types of instructions:

- **speed**—changes the CPU core speed (for saving power)
- **break**—enters debug mode
- **page**—writes to the PA2:0 bits in the STATUS register
- **loadh**/**loadl**—loads a 16-bit pointer into the DPH and DPL registers
- **iread**—reads a word from external memory, program flash memory, or program RAM
- **ireadi**—reads a word (and auto-increments ADDR by 2) from program flash memory, or program RAM
- **iwrite**—writes a word to external memory or program RAM
- **iwritei**—writes a word (and auto-increments ADDR by 2) to program RAM
- **fread**—reads a word from flash program memory
- **fwrite**—writes a word to flash program memory
- **ferase**—erases a block of flash program memory
- **cwdt**—clears the Watchdog Timer

4.3 Instruction Pipeline

An instruction goes through a four-stage pipeline to be executed, as shown in Figure 4-11. The first instruction is fetched from the program memory on the first core clock cycle. On the second clock cycle, the first instruction is decoded and a second instruction is fetched. On the third clock cycle, the first instruction is executed, the second instruction is decoded, and a third instruction is fetched. On the fourth clock cycle, the first instruction's results are written to its destination, the second instruction is executed, the third instruction is decoded, and a fourth instruction is fetched. Once the pipeline is full, instructions are executed at the rate of one per clock cycle.

Stage	Core Cycle 1	Core Cycle 2	Core Cycle 3	Core Cycle 4
Fetch	Instruction 1	Instruction 2	Instruction 3	Instruction 4
Decode		Instruction 1	Instruction 2	Instruction 3
Execute			Instruction 1	Instruction 2
Write				Instruction 1

Figure 4-11 Pipeline Execution

Instructions that directly affect the contents of the program counter (such as jumps and calls) require that the pipeline be cleared and subsequently refilled. Therefore, these instructions take two additional clock cycles (the PC will be changed during the execute cycle of a jump instruction).

4.4 Subroutine Call/Return Stack

A 16-level hardware call/return stack is provided for saving the program counter on a subroutine call and restoring the program counter on subroutine return. The stack is not mapped into the data memory address space except for the top level, which is accessible as the CALLH and CALLL registers. Software can read and write these registers to implement a deeper stack, in those cases which require nesting subroutines more than 16 levels deep. This stack is completely independent of the stack used with the **`push`** and **`pop`** instructions and the SPH/SPL register pair.

Note: The CALLL and CALLH registers require special attention as modification of these values (the top of the stack) changes the return vector.

When a subroutine is called, the return address is pushed onto the subroutine stack, as shown in Figure 4-12. Specifically, each saved address in the stack is moved to the next lower level to make room for the new address to be saved. Stack 1 receives the contents of the program counter. Stack 16 is overwritten with what was in Stack 15. The contents of stack 16 are lost.

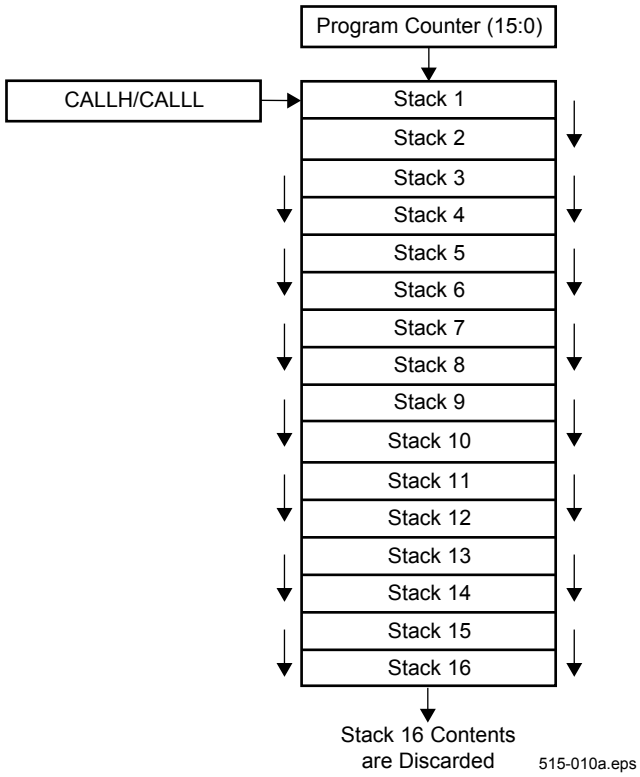


Figure 4-12 Stack Operation on Subroutine Call

When a return instruction is executed the subroutine stack is popped, as shown in Figure 4-13. Specifically, the contents of Stack 1 are copied into the program counter and the contents of each stack level are moved to the next higher level. When a value is popped off the stack, the bottom entry is initialized to 0xFFFF. For example, Stack 1 receives the contents of Stack 2, etc., and Stack 15 is overwritten with the contents of Stack 16. Stack 16 is initialized to 0xFFFF.

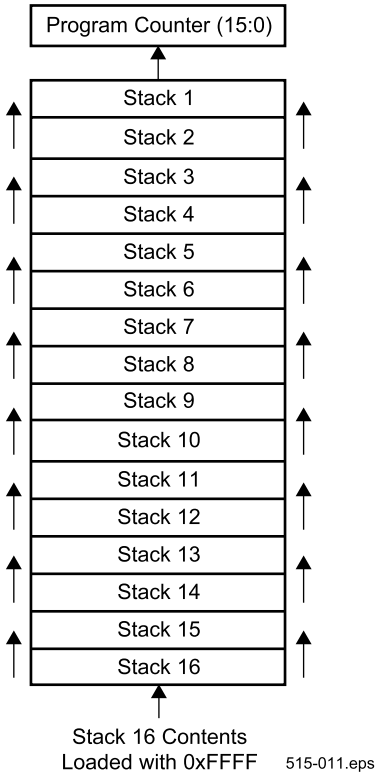


Figure 4-13 Stack Operation on Subroutine Return

For program bugs involving stack underflow, the instruction at byte address 0x1FFFE (word address 0xFFFF) can be used to jump to an appropriate handler. For example, system recovery may be possible by jumping to the reset vector at byte address 0x1FFE0 (word address 0xFFF0).

The options for returning from a CALL are:

1. RET - The stack will be popped (CALLH/L will be loaded into PCH/L) and the page bits (PA2:0 in the STATUS register) will be loaded with the upper 3 bits of CALLH.
2. RETNP - Same as above, but PA2:0 are not changed.
3. RETW #lit - Same as RET, but also moves literal to W.

4.5 Key to Abbreviations and Symbols

Symbol	Description
addr13	13-bit address in assembly language instruction
addr16	16-bit address in assembly language instruction
bit	Bit position selector bit in opcode
BO	Brown-out bit in the PSPCFG register (bit 0)
C	Carry bit in the STATUS register (bit 0)
DC	Digit Carry bit in the STATUS register (bit 1)
DPH	Upper half of data pointer for indirect-with-offset addressing (global file register 0x00C)
DPL	Lower half of data pointer for indirect-with-offset addressing (global file register 0x00D)
f	File register address bit in opcode
fr	File register field (a 9-bit file register address specified in the instruction)
IPH	Indirect Pointer High - Upper half of pointer for indirect addressing (global file register 0x004)
IPL	Indirect Pointer Low - Lower half of pointer for indirect addressing (global file register 0x005)
k	Constant value bit in opcode
n	Numerical value bit in opcode
PA2:PA0	Page bits in the STATUS register (bits 7:5)
PCL	Virtual register for direct PC modification (global file register 0x009)
SPH	Upper half of stack pointer for indirect-with-offset addressing (global file register 0x006)
SPL	Lower half of stack pointer for indirect-with-offset addressing (global file register 0x007)
STATUS	STATUS register (global file register 0x00B)
W	Working register
WD	Watchdog Timeout bit in the PSPCFG register (bit 1)
WDT	Watchdog Timer counter and prescaler
Z	Zero bit in the STATUS register (bit 2
,	File register/bit selector separator (e.g. **`clrb status,z`**)
!=	inequality
#	Immediate literal designator in assembly language instruction (e.g. **`mov w,#0xff`**)
#lit8	8-bit literal value in assembly language instruction
&	Logical AND
(address)	Contents of memory referenced by address
^	Logical exclusive OR
|	Logical OR
||	Concatenation

4.6 Instruction Set Summary Tables

Table 4-2 through Table 4-7 list all of the IP2012 / IP2022 instructions, organized by category. For each instruction, the table shows the instruction mnemonic (as written in assembly language), a brief description of what the instruction does, the number of instruction cycles required for execution, the binary opcode, and the flags in the STATUS register affected by the instruction.

Although the number of clock cycles for execution is typically 1, for the skip instructions the exact number of cycles depends whether the skip is taken or not taken. Taking the skip adds 1 cycle. The effect of extended skip instructions (i.e. a skip followed by a **`loadh`**, **`loadl`**, or **`page`** instruction) is not shown.

For more detailed description, refer to the Programmer's Reference Manual.

Table 4-2 Logical Instructions

Assembler Syntax	Pseudocode Definition	Description	Core Cycles	Opcode	Flags Affected
`and fr,w`	fr = fr & W	AND fr,W into fr	1	`0001 011f ffff ffff`	Z
`and w,fr`	W = W & fr	AND W,fr into W	1	`0001 010f ffff ffff`	Z
`and w,#lit8`	W = W & lit8	AND W,literal into W	1	`0111 1110 kkkk kkkk`	Z
`not fr`	fr = $\overline{fr}$	Complement fr into fr	1	`0010 011f ffff ffff`	Z
`not w,fr`	W = $\overline{fr}$	Complement fr into W	1	`0010 010f ffff ffff`	Z
`or fr,w`	fr = fr | W	OR fr,W into fr	1	`0001 001f ffff ffff`	Z
`or w,fr`	W = W | fr	OR W,fr into W	1	`0001 000f ffff ffff`	Z
`or w,#lit8`	W = W | lit8	OR W,literal into W	1	`0111 1101 kkkk kkkk`	Z
`xor fr,w`	fr = fr ^ W	XOR fr,W into fr	1	`0001 101f ffff ffff`	Z
`xor w,fr`	W = W ^ fr	XOR W,fr into W	1	`0001 100f ffff ffff`	Z
`xor w,#lit8`	W = W ^ lit8	XOR W,literal into W	1	`0111 1111 kkkk kkkk`	Z

Table 4-3 Arithmetic and Shift Instructions

Assembler Syntax	Pseudocode Definition	Description	Core Cycles	Opcode	Flags Affected
`add fr,w`	fr = fr + W	Add fr,W into fr	1	`0001 111f ffff ffff`	C, DC, Z
`add w,fr`	W = W + fr	Add W,fr into W	1	`0001 110f ffff ffff`	C, DC, Z
`add w,#lit8`	W = W + lit8	Add W,literal into W	1	`0111 1011 kkkk kkkk`	C, DC, Z
`addc fr,w`	fr = C + fr + W	Add carry,fr,W into fr	1	`0101 111f ffff ffff`	C, DC, Z
`addc w,fr`	W = C + W + fr	Add carry,W,fr into W	1	`0101 110f ffff ffff`	C, DC, Z
`clr fr`	fr = 0	Clear fr	1	`0000 011f ffff ffff`	Z
`cmp w,fr`	fr - W	Compare W,fr then update STATUS	1	`0000 010f ffff ffff`	C, DC, Z
`cmp w,#lit8`	lit8 - W	Compare W,literal then update STATUS	1	`0111 1001 kkkk kkkk`	C, DC, Z
`cse w,fr`	if (fr - W) = 0 then skip	Compare W,fr then skip if equal	1 or 2 (skip)	`0100 001f ffff ffff`	None
`cse w,#lit8`	if (lit8 - W) = 0 then skip	Compare W,literal then skip if equal	1 or 2 (skip)	`0111 0111 kkkk kkkk`	None
`csne w,fr`	if (fr - W) != 0 then skip	Compare W,fr then skip if not equal	1 or 2 (skip)	`0100 000f ffff ffff`	None
`csne w,#lit8`	if (lit8 - W) != 0 then skip	Compare W,literal then skip if not equal	1 or 2 (skip)	`0111 0110 kkkk kkkk`	None
`cwdt`	WDT = 0	Clear Watchdog Timer	1	`0000 0000 0000 0100`	None
`dec fr`	fr = fr - 1	Decrement fr into fr	1	`0000 111f ffff ffff`	Z
`dec w,fr`	W = fr -1	Decrement fr into W	1	`0000 110f ffff ffff`	Z
`decsnz fr`	fr = fr - 1 if fr != 0 then skip	Decrement fr into fr then skip if not zero (STATUS not updated)	1 or 2 (skip)	`0100 111f ffff ffff`	None

Table 4-3 Arithmetic and Shift Instructions (continued)

Assembler Syntax	Pseudocode Definition	Description	Core Cycles	Opcode	Flags Affected
decsnz w,fr	W = fr - 1 if fr != 0 then skip	Decrement fr into W then skip if not zero (STATUS not updated)	1 or 2 (skip)	**0100 110f ffff ffff**	None
decsz fr	fr = fr - 1 if fr = 0 then skip	Decrement fr into fr then skip if zero (STATUS not updated)	1 or 2 (skip)	**0010 111f ffff ffff**	None
decsz w,fr	W = fr - 1 if fr = 0 then skip	Decrement fr into W then skip if zero (STATUS not updated)	1 or 2 (skip)	**0010 110f ffff ffff**	None
inc fr	fr = fr + 1	Increment fr into fr	1	**0010 101f ffff ffff**	Z
inc w,fr	W = fr + 1	Increment fr into W	1	**0010 100f ffff ffff**	Z
incsnz fr	fr = fr + 1 if fr != 0 then skip	Increment fr into fr then skip if not zero (STATUS not updated)	1 or 2 (skip)	**0101 101f ffff ffff**	None
incsnz w,fr	W = fr + 1 if fr != 0 then skip	Increment fr into W then skip if not zero (STATUS not updated)	1 or 2 (skip)	**0101 100f ffff ffff**	None
incsz fr	fr = fr + 1 if fr = 0 then skip	Increment fr into fr then skip if zero (STATUS not updated)	1 or 2 (skip)	**0011 111f ffff ffff**	None
incsz w,fr	W = fr + 1 if fr = 0 then skip	Increment fr into W then skip if zero (STATUS not updated)	1 or 2 (skip)	**0011 110f ffff ffff**	None
muls w,fr	MULH || W = W x fr	Signed 8 x 8 multiply (bit 7 = sign); W x fr into MULH || W (bit 7 of MULH is result sign)	1	**0101 010f ffff ffff**	None
muls w,#lit8	MULH || W = W x lit8	Signed 8 x 8 multiply (bit 7 = sign); W x literal into MULH || W (bit 7 of MULH is result sign)	1	**0111 0011 kkkk kkkk**	None
mulu w,fr	MULH || W = W x fr	Unsigned 8 x 8 multiply; W x fr into MULH || W	1	**0101 000f ffff ffff**	None
mulu w,#lit8	MULH || W = W x lit8	Unsigned 8 x 8 multiply; W x literal into MULH || W	1	**0111 0010 kkkk kkkk**	None
rl fr	fr || C = C || fr	Rotate fr left through carry into fr	1	**0011 011f ffff ffff**	C
rl w,fr	W || C = C || fr	Rotate fr left through carry into W	1	**0011 010f ffff ffff**	C
rr fr	C || fr = fr || C	Rotate fr right through carry into fr	1	**0011 001f ffff ffff**	C
rr w,fr	C || W = fr || C	Rotate fr right through carry into W	1	**0011 000f ffff ffff**	C
sub fr,w	fr = fr - W	Subtract W from fr into fr	1	**0000 101f ffff ffff**	C, DC, Z
sub w,fr	W = fr - W	Subtract W from fr into W	1	**0000 100f ffff ffff**	C, DC, Z
sub w,#lit8	W = lit8 - W	Subtract W from literal into W	1	**0111 1010 kkkk kkkk**	C, DC, Z
subc fr,w	fr = fr - $\overline{C}$ - W	Subtract $\overline{carry}$,W from fr into fr	1	**0100 101f ffff ffff**	C, DC, Z
subc w,fr	W = fr - $\overline{C}$ - W	Subtract $\overline{carry}$,W from fr into W	1	**0100 100f ffff ffff**	C, DC, Z

Table 4-3 Arithmetic and Shift Instructions (continued)

Assembler Syntax	Pseudocode Definition	Description	Core Cycles	Opcode	Flags Affected
`swap fr`	fr = fr3:0 || fr7:4	Swap high,low nibbles of fr into fr	1	`0011 101f ffff ffff`	None
`swap w,fr`	W = fr3:0 || fr7:4	Swap high,low nibbles of fr into W	1	`0011 100f ffff ffff`	None
`test fr`	if fr = 0 then Z = 1 else Z = 0	Test fr for zero and update Z	1	`0010 001f ffff ffff`	Z

Table 4-4 Bit Operation Instructions

Assembler Syntax	Pseudocode Definition	Description	Core Cycles	Opcode	Flags Affected
`clrb fr,bit`	fr,bit = 0	Clear bit in fr	1	`1000 bbbf ffff ffff`	None
`sb fr,bit`	if fr,bit = 1 then skip	Test bit in fr then skip if set	1 or 2 (skip)	`1011 bbbf ffff ffff`	None
`setb fr,bit`	fr,bit = 1	Set bit in fr	1	`1001 bbbf ffff ffff`	None
`snb fr,bit`	if fr,bit = 0 then skip	Test bit in fr then skip if clear	1 or 2 (skip)	`1010 bbbf ffff ffff`	None

Table 4-5 Data Movement Instructions

Assembler Syntax	Pseudocode Definition	Description	Core Cycles	Opcode	Flags Affected
`mov fr,w`	fr = W	Move W into fr	1	`0000 001f ffff ffff`	None
`mov w,fr`	W = fr	Move fr into W	1	`0010 000f ffff ffff`	Z
`mov w,#lit8`	W = lit8	Move literal into W	1	`0111 1100 kkkk kkkk`	None
`push fr`	(SP) = fr, then SP = SP - 1	Move fr onto top of stack	1	`0100 010f ffff ffff`	None
`push #lit8`	(SP) = lit8, then SP = SP - 1	Move literal onto top of stack	1	`0111 0100 kkkk kkkk`	None
`pop fr`	fr = (SP + 1), then SP = SP + 1	Move top of stack + 1 into fr	1	`0100 011f ffff ffff`	None

Table 4-6 Program Control Instructions

Assembler Syntax	Description	Core Cycles	Opcode	Flags Affected
`call addr13`	Call subroutine	3	`110k kkkk kkkk kkkk`	None
`jmp addr13`	Jump	3	`111k kkkk kkkk kkkk`	None
`int`	Software interrupt	3	`0000 0000 0000 0110`	None
`nop`	No operation	1	`0000 0000 0000 0000`	None
`ret`	Return from subroutine	3	`0000 0000 0000 0111`	PA2:0
`retnp`	Return from subroutine, without updating page bits	3	`0000 0000 0000 0010`	None
`reti #lit3`	Return from interrupt (see Section 3.7.4)	3	`0000 0000 0000 1nnn`	All
`retw #lit8`	Return from subroutine with literal into W	3	`0111 1000 kkkk kkkk`	PA2:0

Table 4-7 System Control Instructions

Assembler Syntax	Description	Core Cycles	Opcode	Flags Affected
`break`	Software breakpoint. Keeps PC from advancing and stops timers, including the Watchdog Timer	1	`0000 0000 0000 0001`	None
`breakx`	Software breakpoint, extending the skip	1	`0000 0000 0000 0101`	None
`ferase`	Erase a 256 word flash block	1†	`0000 0000 0000 0011`	None
`fread`	Read flash memory	1†	`0000 0000 0001 1011`	None
`fwrite`	Write flash memory	1†	`0000 0000 0001 1010`	None
`iread`	Read external/program memory	4(blocking), 1†(nonblocking)	`0000 0000 0001 1001`	None
`ireadi`	Read program memory and increment ADDRL to next even ADDRL	4(blocking), 1†(nonblocking)	`0000 0000 0001 1101`	None
`iwrite`	Write into external memory/program RAM	4(blocking), 1†(nonblocking)	`0000 0000 0001 1000`	None
`iwritei`	Write into program RAM and increment ADDRL to next even ADDRL	4(blocking), 1†(nonblocking)	`0000 0000 0001 1100`	None
`loadh addr8`	Load high data address into DPH	1	`0111 0000 kkkk kkkk`	None
`loadl addr8`	Load low data address into DPL	1	`0111 0001 kkkk kkkk`	None
`page addr3`	Load page bits from program address into PA2:0 of the STATUS register	1	`0000 0000 0001 0nnn`	PA2:0
`speed #lit8`	Change CPU speed by writing into the SPDREG register	1	`0000 0001 nnnn nnnn`	None

†. Only occupies the CPU pipeline for 1 cycle, but the operation is not complete until XCFG:0 = 0. (Refer to Section 4.7)

4.7 Self-Programming and Read Instructions

Table 4-8 Instructions Used for Self-Programming

Operation	Program RAM (ADDRX = 00)	Flash (ADDRX =01)	External Memory (ADDRX = 80 or 81)
Read	**iread** (Blocking) **ireadi** (Blocking)	**fread** (Nonblocking) [1] **iread** [2] **ireadi** [2]	**iread** (Nonblocking)
Write	**iwrite** (Blocking) **iwritei** (Blocking)	**fwrite** (Nonblocking) [3]	**iwrite** (Nonblocking)
Erase	N/A	**ferase** (Nonblocking) [3]	N/A

1 — Rules 1, 2, 3, 5, 7, and 10 below apply.

2 — Rules 2, 3, 5, 7, and 10 below apply. If executed from program RAM, the instruction is nonblocking; if executed from flash, it is blocking.

3 — Rules 1, 2, and 4–10 below apply.

The IP2022 has several instructions used to read and write the program RAM and the program flash memory. These instructions allow the program flash memory to be read and written through special-purpose registers in the data memory space, which allows the flash memory to be used to store both program code and data.

Because no special programming voltage is required to write to the flash memory, any application may take advantage of this feature at run-time. Typical uses include saving phone numbers and passwords, downloading new or updated software, and logging infrequent events such as errors and Watchdog Timer overflow.

The self-programming instructions are not affected by the code-protection flag (the $\overline{CP}$ bit of the FUSE1 register), so the entire program memory is readable and writable by any software running on the IP2012 / IP2022.

Note: It is highly recommended to enable the brown-out reset feature if self-programming instructions are being used in user program code (see Section 3.8.1 for more information about BOR). This will avoid corruption of flash memory during power down.

There are seven instructions used for self-programming, as shown in Table 4-8. Certain uses of the instructions are not valid. In these cases, the instruction is executed as though it were a **nop** instruction (i.e. the program counter is incremented, but no other registers or bits are affected).

Blocking instructions take 4 cycles to complete, and prevent other mainline instructions from executing, but do not block interrupts. Non-blocking instructions occupy the CPU pipeline for only one cycle, but they launch a multi-cycle operation which is not complete until indicated by the FBUSY bit in the XCFG register becoming clear.

The DATAH/DATAL register is a 16-bit data buffer used for loading or unloading data in program memory. The ADDRX/ADDRH/ADDRL register holds a 24-bit byte address used to specify the low-byte of the desired word location in program memory. Like the other pointer registers (IPH/IPL, DPH/DPL, and SPH/SPL), addition to the low byte of the register that results in carry will cause the high part of the register (ADDRX/ADDRH) to be incremented. Subtraction from the low byte of the register that results in borrow will cause the high part of the register to be decremented.

Note: If ADDRSEL is modified in the ISR, it must first be shadowed in software, and restored before **reti**.

Note: ADDRL bit 0 is ignored as the A0 address bit is handled automatically in hardware.

Software should use the FBUSY bit to check that a previous flash write or erase operation has completed before executing another instruction that accesses flash memory, before jumping to or calling program code in flash memory, and before changing the CPU core speed. It is not necessary to check the FBUSY bit if enough cycles are allowed for the flash operation to complete. See description of FRDTS1:0, FRDTC1:0 and FWRT3:0 in Section 7.1.5 for more details. Software must not attempt to execute out of flash memory while the FBUSY bit is set, because the flash memory is unreadable during that time. Therefore, code which reads, writes, or erases flash memory, using the **fread**, **fwrite** or **ferase** instructions, must execute from program RAM. Software must provide at least four cycles between an **fread** and reading DATAH/DATAL, or ensure that the minimum flash read time is met.

Unlike RAM, flash memory requires an explicit erase operation before being written. The **ferase** instruction is used to erase a 512-byte (256-word) block of flash memory (it brings all bits to 1, see Table 4-9). After the block has been erased, individual words can be written with the **fwrite** instruction (**fwrite** will not change a 0 to 1). For example, an **ferase** instruction executed on any byte address from 0x10000 to 0x100FE erases the whole block spanning those addresses. The self-programming instructions have no access to the flash memory bits in the configuration block.

Table 4-9 ferase Addresses (ADDRX=01, ADDRL=xx)

ADDRH	Flash Byte Addresses
0x00	0x10000 - 0x101FE
0x02	0x10200 - 0x103FE
...	...
0xFE	0x1FE00 - 0x1FFFE

Rules/Troubleshooting for **fread**/**fwrite**/**ferase** and **iread**/**ireadi** of flash:

1. Must be executing out of program RAM, with ADDRX = 01.
2. FCFG register must be correctly configured (refer to Section 7.1.5).
3. For an **fread** or **iread**/**ireadi** of flash, there must be at least 3 core cycles between the read instruction and a read of DATAH or DATAL, or the minimum flash read time must be met.
4. No speed commands while **fread**, **fwrite** or **ferase** are busy (while XCFG bit 0 = 1).
5. Do not jump to flash memory while executing **fread**/**fwrite**/**ferase**. If INTVEC is in flash, ensure interrupts are disabled.
6. **fwrite** will not change a 0 to a 1 (use **ferase** first).
7. XCFG bit 0 = 0 before execution (even during ISR) or sufficient time is allowed to complete previous operations on flash.
8. XCFG bit 6 = 1, otherwise **fwrite** and **ferase** behave as **nop**.
9. Make sure interrupts are disabled or that the INTSPD value matches the SPDREG value. For **iread** or **ireadi** of flash from flash, a more practical solution is to jump to a routine in program RAM.
10. Wait 1 cycle after changing ADDRX bit 7, EMCFG bit 7, or ADDRSEL before executing an **fread**, **fwrite**, **ferase**, **iread**, **ireadi**, **iwrite**, or **iwritei** instruction. Exception: This one-cycle wait is not necessary if the software is not switching between memory blocks (if ADDR is not changing between Program RAM, Flash, or External Memory), or if an ADDRSEL change does not cause a memory block change.
11. Do not write to DATAH at the same time an **iread** of External Memory is causing a write of DATAL.

4.7.1 Flash Timing Control

The FCFG register controls the timing of flash memory operations. See Section 7.1.5 for a description of the FCFG register.

4.7.2 Interrupts During Flash Operations

Before starting a flash write or erase operation, the FCFG register (see Section 7.1.5) must be set up properly for the current speed. The CPU core clock is the time base for the flash write timing compensation, so it is critical that the CPU core clock speed is not changed during a flash write or erase operation. Interrupts may be taken during a flash write or erase operation, if the INTSPD register is set up so the speed does not change when an interrupt occurs.

If the flash read timing compensation is set up for a clock divisor of 1 (i.e. fastest speed), interrupts will not cause **fread**/**iread** instructions to fail, so no special precautions need to be taken to avoid violating the flash read access time.

5.0 Peripherals

The IP2022 provides an array of on-chip peripherals needed to support a broad range of embedded Internet applications:

- 2 Serializer/Deserializer (SerDes) units
- Real-time timer
- T0 timer
- 2 General-purpose timers with compare and capture Registers
- Watchdog timer
- 10-bit, 8-channel A/D converter
- Analog comparator
- Parallel slave peripheral interface
- External memory interface

All of the peripherals except the Watchdog Timer and the Real-Time Timer use alternate functions of the I/O port pins to interface with external signals.

5.1 I/O Ports

The IP2022 contains one 4-bit I/O port (Port A) and six 8-bit I/O ports (Port B through Port G). The four Port A pins have 24 mA current drive capability. All the ports have symmetrical drive. Inputs are 5V-tolerant. Outputs can use the same 2.3–2.7V power supply used for the CPU core and peripheral logic, or they can use a higher voltage (up to 3.6V). The IOVdd pins are provided for the I/O port pin output drivers. Port G has a separate GVdd pin which can be used to run the Port G output drivers at a voltage different from that used for the other ports, since Port G must run from a 2.3–2.7V power supply.

Each port has separate input (RxIN), output (RxOUT), and direction (RxDIR) registers, which are memory mapped. The numbers in the pin names correspond to the bit positions in these registers. These registers allow each port bit to be individually configured as a general-purpose input or output under software control. Unused pins should be configured as outputs, to prevent them from floating. Port B has three additional registers for supporting external interrupts (see Section 5.1.1).

Each port pin has an alternate function used to support the on-chip hardware peripherals, as listed in Table 2-1. Port A and Port B support the multi-function timers Timer 1 and Timer 2. Port B, Port C, and Port D support the Parallel Slave Peripheral (PSP) and external memory functions. Port E and Port F support the serializer/deserializer (SERDES) units. Port G supports the analog to digital converter (ADC) and the analog comparator. Before enabling a hardware peripheral, configure the port pins for input or output as required by the peripheral.

Note: There is positive-feedback circuitry present on the I/O ports when configured as input. This causes an input that was previously high, then subsequently tri-stated (i.e. not driven), to be actively driven by the IP2012 / IP2022 to one diode drop below DVdd. See Section 8.2 for details.

Figure 5-1 shows the internal hardware structure and configuration registers for each pin of a port.

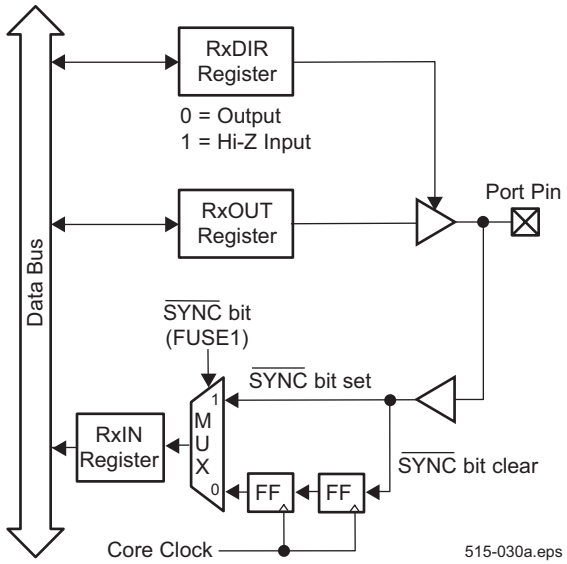


Figure 5-1 Port Pin Block Diagram

5.1.1 Port B Interrupts

Any of the 8 Port B pins can be configured as an external interrupt input. Logic on these inputs can be programmed to sense rising or falling edges. When an edge is detected, the interrupt flag for the port pin is set.

The recommended initialization sequence is:

1. Configure the port pins used for interrupts as inputs by programming the RBDIR register.
2. Be sure all enabled interrupt pins are driven to valid logic levels, not floating.
3. Select the desired edge for triggering the interrupt by programming the INTED register. This may set interrupt flags.
4. `nop`, `nop`.
5. Clear the interrupt flags in the INTF register.
6. `nop`
7. Enable the interrupt input(s) by setting the corresponding bit(s) in the INTE register.
8. Set the GIE bit.

Figure 5-2 shows the Port B interrupt logic. Port B has three registers for supporting external interrupts, the INTED (Section 5.1.6), INTF (Section 5.1.7), and INTE

(Section 5.1.8) registers. The INTED register controls the logic which selects the edge sensitivity (i.e. rising or falling edge) of the Port B pins. When an edge of the selected type occurs, the corresponding flag in the INTF register is set, whether or not the interrupt is enabled. The interrupt signal passed to the system interrupt logic is the OR function of the AND of each interrupt flag in the INTF register with its corresponding enable bit in the INTE register. See Section 5.1.8.

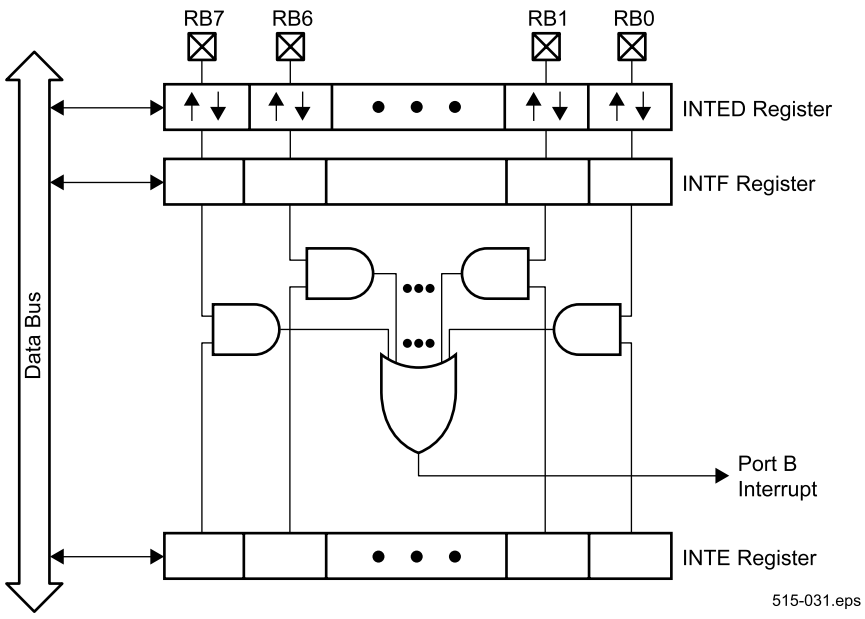


Figure 5-2 Port B Interrupt Logic

5.1.2 Reading and Writing the Ports

The port registers are memory-mapped into the data memory address space between 0x020 and 0x03A. In addition, Port B has three extra registers located at 0x017 through 0x019 (INTED, INTF, and INTE), which support external interrupt inputs.

Generally, successive read and write operations on the same I/O port is not an issue, as there are separate IN an OUT registers for each I/O port. Care must be given to ensure that enough time is allowed for data written to the OUT register to propagate to the IN register on a given port. If this is an issue, two instructions (or four instructions if the SYNC bit in the FUSE1 register is clear) should be inserted between any read-modify-write instruction sequences (or more **nop** instructions if the pin is capacitively loaded).

5.1.3 RxIN Registers

The RxIN registers are virtual registers that provide read-only access to the physical I/O pins. Reading these registers returns the states on the pins, which may be driven either by the IP2022 or an external device. If the SYNC bit in the FUSE1 register is clear, the states are read from a synchronization register. If an application reads data from a device running asynchronously to the IP2022, the SYNC bit should be cleared to avoid the occurrence of metastable states (i.e. corrupt data caused by an input which fails to meet the setup time before the sampling clock edge, which theoretically could interfere with the operation of the CPU).

5.1.4 RxOUT Registers

The RxOUT registers are data output buffer registers. The data in these registers is driven on any I/O pins that are configured as outputs. On reads, the RxOUT registers return the data previously written to the data output buffer registers, which might not correspond to the states actually present on pins configured as inputs or pins forced to another state by an external device.

5.1.5 RxDIR Registers

The RxDIR registers select the direction of the port pins. For each output port pin, clear the corresponding RxDIR bit. For each input port pin, set the corresponding RxDIR bit. Unused pins that are left open-circuit should be configured as outputs, to keep them from floating.

For example, to configure Port A pins RA3 and RA2 as outputs and RA1 and RA0 as inputs, the following code could be used:

```
mov    w,#0x03     ;load W with the value 0x03
                   ;(bits 3:2 low, and bits 1:0
                   ;high)
mov    0x022,w     ;write 0x03 to RADIR
                   ;register
```

The second move instruction in this example writes the RADIR register, located at address 0x022. Because Port A has only four I/O pins, only the four least significant bits of this register are used.

To drive the RA1 pin low and the RA0 pin high, the following code then could be executed:

```
mov    w,#0x01     ;load W with the value 0x01
                   ;(bits 3:1 low, and bit 0
                   ;high)
mov    0x021,w     ;write 0x01 to RAOUT
                   ;register
```

The second move instruction shown above writes the RAOUT register, located at address 0x021. When reading the Port A pins through the RAIN register (0x020), the upper four bits always read as zero.

When a write is performed to the RxOUT register of a port pin that has been configured as an input, the write is performed but it has no immediate effect on the pin. If that pin is later configured as an output, the pin will be driven with the data that had been previously written to the RxOUT register.

5.1.6 INTED Register

The INTED register consists of 8 edge detection bits that correspond to the 8 pins of Port B. A set bit in the INTED register makes the corresponding port pin trigger on falling edges, while a clear bit makes the pin trigger on rising edges.

5.1.7 INTF Register

The INTF register consists of 8 interrupt flags that correspond to the 8 pins of Port B. If the trigger condition for a Port B pin occurs, the corresponding bit in the INTF register is set. The bit is set even if the port pin is not enabled as a source of interrupts.

The interrupt service routine (ISR) can check this register to determine the source of an external interrupt. If a Port B pin enabled for generating interrupts has a set bit in the INTF register, software must clear the bit prior to exiting to prevent repeated calls to the ISR.

The Port B interrupt logic is asynchronous (e.g. functions without a clock in clock-stop mode). A side effect is that there is a 2-cycle delay between the instruction that clears a INTF bit and the bit being cleared. This means that software must clear the bit at least 2 cycles before executing a return from interrupt (**`reti`**) instruction.

5.1.8 INTE Register

The INTE register consists of 8 interrupt enable bits that correspond to the 8 pins of Port B. A Port B pin is enabled as a source of interrupts by setting the corresponding bit in the INTE register. The pin is disabled as an interrupt source by clearing the corresponding INTE bit, but takes up to 1 core clock cycle for the interrupt to be disabled.

5.1.9 Port Configuration Upon Power-Up

On power-up, all the port control registers (RxDIR) are initialized to 0xFF. Therefore, each port pin is configured as a high-impedance input. This prevents any false signalling to external components which could occur if the ports were allowed to assume a random configuration at power-up.

5.2 Timer 0

Timer 0 is an 8-bit timer with an 8-bit prescaler intended to generate periodic interrupts for ipModule™ instances that require being called at a constant rate, such as UART and DTMF functions. When the T0TMR register counts up to FF and rolls over to 00, the T0IF flag in the T0CFG register will be set, and an interrupt will occur if the T0IE and T0EN bit are set (see T0CFG register description in Section 7.1.20). To clear the interrupt, either the T0IE or T0EN bit should be cleared, and then the T0IF flag must be cleared.

Note: If T0IF is not cleared after disabling the Timer0 interrupt (T0IE = 0) or disabling Timer0 (T0EN = 0), it is assumed that another interrupt has occurred, and the interrupt will occur on the next return, or when GIE is set (enabling nested interrupts - see Section 3.7.2).

The Timer 0 interrupt is also supported in the instruction set by an option for the **`reti`** instruction which adds the W register to the T0TMR register when returning from an interrupt. Figure 5-3 shows the Timer 0 logic.

Operation of Timer 0 to generate periodic interrupts:

- T0TMR = 00 when entering ISR from T0 interrupt
- Keeps counting up while in ISR
- Add W to T0TMR with execution of **`reti`** (refer to Table 3-5). Interrupt frequency is adjusted by

adjusting value loaded in W, and depending on core clock divider, since T0TMR runs on the system clock. If W added to T0TMR exceeds 0xFF, no interrupt is taken until the T0TMR rolls over from 0xFF to 0x00 again. If the T0TMR rolls over during the 3 core cycles in the return from interrupt, the ISR is executed again (and never again returns to mainline code as long as the ISR executes the same).

Note: Do not enable Timer 0 interrupt before enabling the Timer 0 itself.

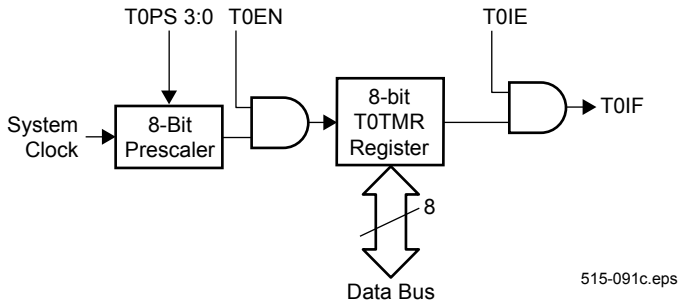


Figure 5-3 Timer 0 Block Diagram

The control and status register for Timer 0 is the T0CFG register, described in detail in Section 7.1.20.

Note: T0IF can only be asserted when T0IE = 1, T0EN = 1 and T0TMR overflow occurs.

5.3 Real-Time Timer (RTTMR)

The Real-Time Timer is an 8-bit timer intended to provide a periodic system wake-up interrupt. Unlike the other peripherals (except the Watchdog Timer and Port B interrupts), the Real-Time Timer continues to function when the system clock is disabled. For those applications which spend much of their time with the OSC clock oscillator turned off to conserve power, there are 5 available mechanisms to exit this mode: external reset ($\overline{\text{RST}}$ pin), reset from the Watchdog Timer, reset from Brown-out, interrupt from a Port B input, and interrupt from the Real-Time Timer. By using an interrupt rather than reset, more of the CPU state is preserved and some reset procedures such as initializing the port direction registers can be skipped. Figure 5-4 shows the Real-Time Timer logic. (When RTCLK1 is not used, it should be tied to GND.)

When the RTTMR register counts up to FF and rolls over to 00, the RTIF flag in the RTCFG register will be set, and an interrupt will occur if the RTIE and RTEN bit are set (see RTCFG register description in Section 7.1.9). To clear the interrupt, either the RTIE or RTEN bit should be cleared, and then the RTIF flag be cleared.

Note: A **`nop`** is required between a **`speed`** instruction and an instruction that enables, reads, or writes RTTMR.

Note: If RTIF is not cleared after disabling the Real-Time Timer interrupt (RTIE = 0) or disabling the Real-Time Timer (RTEN = 0), it is assumed that another interrupt has occurred, and the interrupt will occur on the next return, or when GIE is set (enabling nested interrupts - see Section 3.7.2).

Note: The system clock must be slower or equal to the RTCLK clock, for a write to the RTTMR to work correctly.

The real-time timer is readable and writable as the RTTMR register. The control and status register for the timer is the RTCFG register, as described in Section 7.1.9.

The RTEOS bit (XCFG bit 5, see Section 7.1.26) selects the sampling mode for the external input. If the RTEOS bit is set, the external input is over-sampled with the system clock. The CPU can always read the value in the RTTMR register, if the system clock is at least twice the frequency of the external input. If the system clock source is changed to RTCLK or turned off, then the RTEOS bit must be clear for the Real-Time Timer to function.

Note: if the RTEOS bit is cleared, expect a 3 cycle system clock delay for the overflow interrupt, due to synchronization circuitry.

If the RTEOS bit is clear then the external input directly clocks the Real-Time Timer (i.e. RTCLK is not oversampled). The Real-Time Timer will always function whether the clock input is synchronous or asynchronous. However, the CPU cannot reliably read the value in the RTTMR register unless the RTCLK clock is synchronous to the system clock (RTEOS=1).

If the value in the RTTMR register does not need to be used by the CPU (i.e. only the interrupt flag is of interest), then the RTEOS bit should be clear (i.e. RTCLK not oversampled), which allows the Real-Time Timer to function for any configuration of the system clock.

If the value in the RTTMR register needs to be used by the CPU, but the Real-Time Timer is not required to function when the system clock is set to RTCLK or turned off, then the RTEOS bit should be set to ensure the CPU can reliably read the RTTMR register.

If the value in the RTTMR register needs to be used by the CPU and the Real-Time Timer is required to function when the system clock is set to RTCLK or off, then software must change the RTEOS bit when changing the system clock source. To read the RTTMR register when the system clock is not synchronous to the RTCLK, the RTEOS bit must be set to ensure reliable operation. Before the system clock is changed to RTCLK or turned off, the RTEOS bit must be clear (i.e. RTCLK not

oversampled) for the Real-Time Timer to continue to function.

Note: When using development tools in single stepping mode, the RTSS bit must be cleared and RTEOS must be set, otherwise the counter will behave erratically.

Note: Care must be exercised if Port B interrupts and RTTMR interrupts are enabled, because the RTTMR may receive sporadic clocks during crystal startup while the system clock is waiting for WUDX2:0 (see Figure 3-16).

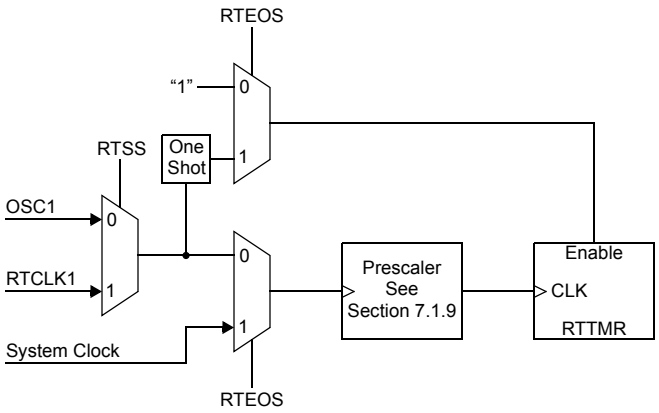


Figure 5-4 Real-Time Timer Block Diagram

5.4 Multi-Function Timers (T1 and T2)

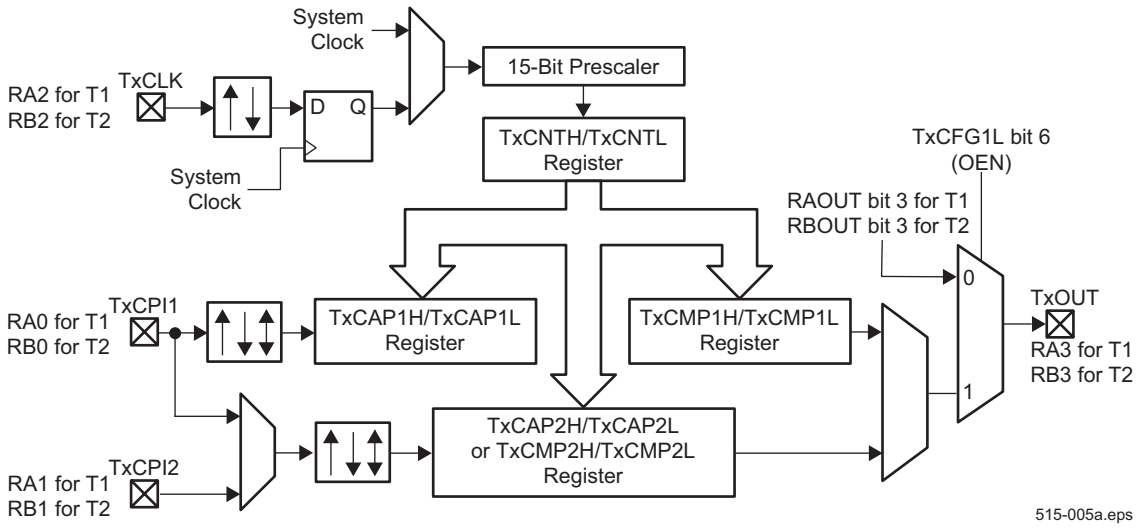


Figure 5-5 Multifunction Timer Block Diagram

The IP2012 / IP2022 contains two independent 16-bit multi-function timers, called T1 and T2 (notated below as Tx). These versatile, programmable timers reduce the software burden on the CPU in real-time control applications such as PWM generation, motor control, triac control, variable-brightness display control, sine-wave generation, and data acquisition.

Each timer consists of a 16-bit counter register supported by a dedicated 16-bit capture register and two 16-bit compare registers. The second compare register can also serve as capture register. Each timer may use up to four external pins: TxCPI1 (Capture Input), TxCPI2 (Capture Input), TxCLK (Clock Input), TxOUT (Output). These pins are multiplexed with general-purpose I/O port pins. The port direction register has priority over the timer configuration, so the port direction register must be programmed appropriately for each of these four signals if their associated timer functions are used.

Figure 5-5 is a block diagram showing the registers and I/O pins of one timer. Each timer is based on a 16-bit counter/timer driven by a 15-bit prescaler. The input of the prescaler can be either the system clock or an external clock signal which is internally synchronized to the system clock. The counter cannot be directly written by software, but it may be cleared by writing to the TxRST bit in the TxCTRL register.

5.4.1 Timers T1, T2 Operating Modes

Each timer can be configured to operate in one of the following modes:

- Pulse-Width Modulation (PWM)
- Timer
- Capture/Compare

PWM Mode

In PWM Mode, the timer can generate a pulse-width modulated signal on its output pin, TxOUT. The period of the PWM cycle (high + low), in number of system clocks, is specified by the value in the TxCAP2H/TxCAP2L register. The high time of the pulse is specified by the value in the TxCMP1H/TxCMP1L register.

PWM mode can be used to generate an external clock signal that is synchronous to the IP2012 / IP2022 system clock. For example, by loading TxCMP1H/TxCMP1L with 1 and TxCAP2H/TxCAP2L with 2 (the high registers must be written last for this to work), a symmetric external clock can be generated at ∫ the frequency of the system clock. In some applications, this can eliminate crystals or oscillators required to produce clock signals for other components in the system. SerDes GPSI mode can also produce clock outputs.

The 16-bit counter/timer counts upward, starting with the TxOUT output driven high. After reaching the value stored in the TxCMP1H/TxCMP1L register minus one, at the next clock edge the TxOUT pin is driven low. The counter/timer is unaffected by this event and continues to increment. After reaching the value stored in the TxCAP2H/TxCAP2L register minus one, at the next clock edge the timer is cleared. When the counter is cleared, the TxOUT output is driven high, *unless* the TxCMP1H/TxCMP1L register is clear, in which case the TxOUT pin is driven low.

There are two special cases. When the TxCMP1H/TxCMP1L register is clear, the TxOUT pin is driven with a continuous low, corresponding to a duty-cycle of 0%. When the value in the TxCMP1H/TxCMP1L register is equal to the value in the TxCAP2H/TxCAP2L register, the TxOUT output is driven with a continuous high, corresponding to a duty-cycle of 100%.

The behavior of the timers is undefined when the value in the TxCMP1H/TxCMP1L register is greater than the value in the TxCAP2H/TxCAP2L register.

The timer is glitch-free no matter when the TxCMP1H/TxCMP1L register or the TxCMP2H/TxCMP2L register are changed relative to the value of the internal counter/timer. The new duty cycle or period values do not take effect until the current PWM cycle is completed (the counter/timer is reset).

Interrupts, if enabled through the TxCFG1H register, can be generated whenever the timer output is set or cleared. If the TxCMP1H/TxCMP1L register is clear, or if the value in the TxCMP1H/TxCMP1L register is equal to the value in the TxCAP2H/TxCAP2L register, an interrupt can be generated each time the counter/timer is reset to zero.

In PWM mode, the Capture 1 input remains active (if enabled by the CPI1EN bit in the TxCFG1L register) and, when triggered, captures the current counter/timer value into the TxCAP1 register.

The multifunction timers can be configured to interrupt on a Capture 1 event and reset the counter/timer on the event. For PWM operation without Capture 1, software must disable the Capture 1 input by clearing the CPI1EN bit in the TxCFG1L register.

Timer Mode

This is not a separate timer mode (from the hardware point of view), but is a conceptual mode for programmers. It is the PWM mode, except that software disables the timer output by clearing the OEN bit in the TxCFG register.

Capture/Compare Mode

In Capture/Compare mode, one or both of the timer capture inputs (TxCPI1 and TxCPI2) may be used. Their pin functions must be enabled in the TxCFG1 register. Each capture input can be programmed in the TxCFG2 register to trigger on a rising edge, falling edge, or both rising and falling edges.

When a trigger event occurs on either capture pin, the current value of the counter/timer is captured into the TxCAP1H/TxCAP1L register or the TxCAP2H/TxCAP2L register for that input pin.

The counter/timers can also be configured to reset on a TxCPI1 input event, in which case the value of the counter/timer before it was reset is captured in the TxCAP1H/TxCAP1L register and the counter/timer is reset to zero. This mode is useful for measuring the frequency (or width) of external signals. By using both capture inputs and configuring them for opposite edges, the duty cycle of a signal can also be measured. To avoid wasting I/O port pins in this configuration, the CPI2EN bit in the TxCFG1L register is provided to internally tie the TxCPI1 and TxCPI2 inputs together, which frees the TxCPI2 pin to be used as a general-purpose I/O port pin.

An interrupt can be generated for any capture event and for counter/timer overflows.

This mode also features an output-compare function. The TxCMP1H/TCMP1L register is constantly compared against the internal counter/timer. When the counter/timer reaches the value of the TxCMP1H/TxCMP1L register minus one, at the next counter clock the TxOUT output is toggled. The TxOUT output, if enabled via the OEN bit, can be driven high or low by writing to the TOUTSET and TOUTCLR bits in the TxCFG2L register. An interrupt can be enabled for this event.

Interrupts

When a Multi-Function Timer interrupt occurs, the corresponding interrupt flag (depending on the mode; OFIF, CAP2IF/CMP2IF, CAP1IF or CMP1IF) in the TxCFG1H register will be set, and an interrupt will occur if the TMREN bit (TxCFG1L register), the TxIE bit (TCTRL register) and an interrupt source is enabled (depending on the mode; OFIE, CAP2IE/CMP2IE, CAP1IE or CMP1IE) are set (TxCFG1H register). To clear the interrupt, either the TMREN bit, TxIE bit or the interrupt source (OFIE, CAP2IE/CMP2IE, CAP1IE or CMP1IE) should be cleared, and then the interrupt flag (OFIF, CAP2IF/CMP2IF, CAP1IF or CMP1IF) should be cleared.

Note: The interrupt flag can only be asserted when the multi-function timers are enabled, the timer interrupts are enabled, an interrupt source is enabled, and timer event occurs.

Note: If the interrupt flag is not cleared after disabling either the interrupt enable or the Multi-Function Timer enable (TMREN = 0), it is assumed that another interrupt has occurred, and the interrupt will occur on the next return, or when GIE is set (enabling nested interrupts - see Section 3.7.2).

5.4.2 T1 and T2 Timer Pin Assignments

The following table lists the I/O port pins associated with the Timer T1 and Timer T2 I/O functions.

Table 5-1 Timer T1/T2 Pin Assignments

I/O Pin	Timer T1/T2 Function
RA0	Timer T1 Capture 1 Input
RA1	Timer T1 Capture 2 Input
RA2	Timer T1 External Event Clock Source
RA3	Timer T1 Output
RB0	Timer T2 Capture 1 Input
RB1	Timer T2 Capture 2 Input
RB2	Timer T2 External Event Clock Source
RB3	Timer T2 Output

5.4.3 T1 and T2 Timer Registers

Each timer has six 16-bit register pairs, which are accessed as 8-bit registers in the special-purpose register space. There is also one 8-bit register shared by both timers.

TxCNTH/TxCNTL Register

The TxCNTH/TxCNTL register indicates the value of the counter/timer and increments synchronously with the rising edge of the system clock. This register is read-only. The timer counter may be cleared by writing to the TxRST bit in the TCTRL register.

Reading the TxCNTL register returns the least-significant 8 bits of the internal TxCNT counter and causes the most-significant 8 bits of the counter to be latched into the TxCNTH register. This allows software to read the TxCNTH register later and still be assured of atomicity.

Note: In addition to reading T1CNTL, reading data RAM address $C5 will also cause the most significant 8 bits of the T1 counter to be latched into the T1CNTH register. In addition to reading T2CNTL, reading data RAM address $D5 will also cause the most significant 8 bits of the T2 counter to be latched into the T2CNTH register. Writing to $C5 or $D5 will have no affect on TxCNTH registers.

TxCAP1H/TxCAP1L Register

The TxCAP1H/TxCAP1L register captures the value of the counter/timer when the TxCPI1 input is triggered. This register is read-only.

Reading the TxCAP1L register returns the least-significant 8 bits of an internal capture register and causes the most-significant 8-bits of the register to be latched into the TxCAP1H register. This allows software to read the TxCAP1H register later and still be assured of atomicity.

TxCMP1H/TxCMP1L Register

In Capture/Compare mode, the TxOUT output pin is toggled (if enabled by the OEN bit in the TxCFG1 register) when the counter/timer increments to the value in the TxCMP1 register. In this mode, the value written to the TxCMP1 register takes effect immediately.

Writing to the TxCMP1L register causes the value to be stored in the TxCMP1L register with no other effect. Writing to the TxCMP1H register causes an internal compare register to be loaded with a 16-bit value in which the low 8 bits come from the TxCMP1L register and high 8 bits come from the value being written to the TxCMP1H register. Software should write the TxCMP1L register before writing the TxCMP1H register, because writing to the TxCMP1H register is used as an indication that a new compare value has been written. Writing to the TxCMP1H register is required for the new compare value to take effect - this means that TxCMP1H must be written AFTER TxCMP1L for the value to have any effect. In PWM mode, the 16-bit number latched into the internal compare register by writing to the TxCMP1H register does not take effect until the end of the current PWM cycle.

Reading the TxCMP1H or TxCMP1L registers returns the previously written value whether or not the value stored in these registers has been transferred to the internal compare register by writing to the TxCMP1H register.

TxCAP2H/TxCAP2L or TxCMP2H/TxCMP2L Register

This register may be called the TxCAP2H/TxCAP2L register or TxCMP2H/TxCMP2L register.

In PWM mode, this register determines the period of the PWM signal. In this mode, this register is both readable and writable. However, on writes the value is not applied until the end of the current PWM cycle.

Writing to the TxCAP2L register causes the value to be stored in the TxCAP2L register with no other effect. Writing to the TxCAP2H register causes an internal compare register to be loaded with a 16-bit value in which the low 8 bits come from the TxCAP2L register and the high 8 bits come from the value being written to the TxCAP2H register. Software should write the TxCAP2L

register before writing the TxCAP2H register, because writing to the TxCAP2H register is used as an indication that a new compare value has been written. Writing to the TxCAP2H register is required for the new compare value to take effect. In PWM mode, the 16-bit number latched into the internal compare register by writing to the TxCAP2H register does not take effect until the end of the current PWM cycle.

Reading the TxCAP2H or TxCAP2L registers returns the previously written value regardless of whether the value stored in these registers has been transferred to the internal compare register by writing to the TxCAP2H register.

In Capture/Compare mode, this register captures the value of the counter/timer when the TxCPI2 input is triggered. In this mode, this register is read-only.

Reading the TxCAP2L register returns the least-significant 8 bits of an internal capture register and causes the most-significant 8-bits to be latched into the TxCAP2H register. This allows software to read the TxCAP2H register later and still be assured of atomicity.

TxCFG1H/TxCFG1L Register

Selects timer operation mode, pin functions, interrupts and other configuration settings. See Section 7.1.21 for the description of TxCFG1H and Section 7.1.23 for the description of TxCFG1L.

TxCFG2H/TxCFG2L Register

Selects capture input trigger edges, prescaler setting, and other configuration settings. See Section 7.1.22 for the description of TxCFG2H and Section 7.1.24 for the description of TxCFG2L.

TCTRL Register

Unlike the other timer control registers, one TCTRL register is used to synchronize both timers. Setting the TxRST bit clears the TxCNTH/TxCNTL register pair and the prescaler counter, which allows global synchronization of all timers on the device. There are also individual timer interrupt-enable bits. See Section 7.1.25 for description.

5.5 Watchdog Timer (WDT)

A Watchdog Timer (WDT) is available for recovering from unexpected system software hang-ups. When the Watchdog Timer is enabled, software must periodically clear the timer by executing a **`cwdt`** instruction. Otherwise, the timer will overflow, which resets the IP2012 / IP2022 but doesn't clear the WD bit in the PSPCFG register (this bit should be set before the first **`cwdt`** instruction is executed). Any other source of reset clears the WD bit, so software can use this bit to identify a reset caused by the Watchdog Timer. The Watchdog Timer is shown in Figure 5-6.

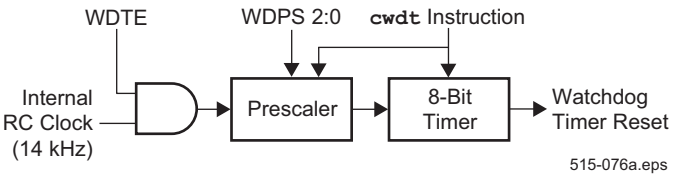


Figure 5-6 Watchdog Timer

The Watchdog Timer is enabled by setting the WDTE bit in the FUSE1 register. The time period between coming out of reset or clearing the timer and timer overflow is controlled by the WDPS2:0 bits in the FUSE1 register, as discussed in Section 3.10.2.

Since the watchdog timer period varies by up to 50% over temperature and voltage, the minimum timeout period selected in FUSE1 that works in nominal conditions, should not be used. For instance, if the 640ms setting works in nominal conditions, the 1280ms setting should be used in production.

The Watchdog Timer register is not visible to software. The only feature of the Watchdog Timer visible to software is the WD bit in the PSPCFG register (see Section 7.1.8).

Note: When using the development tools, the watchdog timer is disabled while in debug mode, except when "Run" command is issued. The **`break`** and **`breakx`** instructions suspend the Watchdog Timer, so that debug mode works correctly. Therefore, if the watchdog feature is used, the **`break`** and **`breakx`** instructions should not be used, and the program RAM should be initialized to instructions that do not include **`break`** and **`breakx`**.

5.6 Serializer/Deserializer (SERDES)

There are two SERDES units in the IP2022, which support a variety of serial communication protocols, including GPSI, SPI, UART, USB, and 10Base-T Ethernet. By performing data serialization/deserialization in hardware, the CPU bandwidth needed to support serial communication is greatly reduced, especially at high baud rates. Providing two units allows easy implementation of protocol conversion or bridging functions between the two high-speed serial interfaces, such as a USB to 10Base-T Ethernet bridge.

Each SERDES unit uses up to 8 external digital signals: SxCLK, SxRXD, SxRXM, SxRXP, SxTXM, SxTXME, SxTXP, and SxTXPE/SxOE. The signals for SERDES1 are multiplexed with the Port E pins, and the signals for SERDES2 are multiplexed with the Port F pins. The port direction bits must be set appropriately for each pin that is used. Not all signals are used in all protocol modes. See Table 5-3 for details on signal port pin usage in various protocol modes. In addition to the digital signals, there are also two analog signals only used in 10Base-T Ethernet mode: SxRX+ and SxRX-.

Note: Proper operation of the SERDES requires that the core-clock be present — don't turn off core clock while SERDES is still transmitting.

SERDES Configuration Registers

The descriptions for the SxMODE, SxRSYNC, SxSMASK, SxRCFG, SxRCNT, SxTCFG, SxTMRH/SxTMRL and SxINTE/SxINTF registers can be found in Section 7.1.

Note: A one cycle delay is required between consecutive read-modify-write instructions to the same SERDES register (for example, **`clrb reg`** and **`setb reg`**).

5.6.1 SERDES TX/RX Buffers

SxRBUFH/SxRBUFL Registers

This is a 16-bit register pair for unloading received data. The RXBF bit in the SxINTF register indicates when new data has been loaded into this register. If the corresponding bit in the SxINTE register is set, an interrupt is generated.

SxTBUFH/SxTBUFL Registers

This is a 16-bit register pair for loading data to be transmitted. The TXBE bit in the SxINTF register indicates when the data has been transmitted and the register is ready to be loaded with new data. If the corresponding bit in the SxINTE register is set, an interrupt is generated.

Regardless of the number of bits to be transmitted, both the high and low registers need to be written to initiate transmission. The order of writing to high or low doesn't matter.

5.6.2 SERDES Configuration

Software prepares a SERDES unit to receive data by programming the receive shift count register (SxRCFG) and the clock select bits in the SxMODE register appropriately for the selected protocol. The SxRCFG register is copied to an internal counter, and when that number of bits of data has been received, the received data is loaded into the SxRBUF register.

In 10Base-T, GPSI, or USB mode, when an EOP is detected, the SxRCNT register is loaded with the number of bits actually received, the EOP bit of the SxINTF register is set, and the data bits are loaded into the SxRBUF register. The RXBF bit in the SxINTE register can be set to enable an interrupt on this event.

The SxTXP and SxTXM pins correspond to the differential outputs of the USB or Ethernet bus. Other serial protocols require only one output pin, which is SxTXP by default.

The SxTXP and SxTXM pins have high current outputs for driving Ethernet magnetics directly without the use of transceivers.

When the clock select register is programmed with the value for 10Base-T, the transmit pre-emphasis requirement enables the SxTXPE and SxTXME outputs, which have a 50ns-delayed version of the transmit output that is resistively combined outside the chip before driving the magnetics.

For transmitting, software must specify the number of bits to transmit (specified in the SxTCFG register) and load the data in the SxTBUF register. This data is then transferred to an internal register, from which it is serially shifted out to the transmit logic. The TXBE bit in the SxINTE register can be set to enable an interrupt when the data has been transferred from the SxTBUF register. When there is a transmit buffer underrun event (i.e. all of the data has been shifted out from the internal register, but the SxTBUF register has not been reloaded), an EOP condition is generated on the SxTXP and SxTXM outputs after an internal counter decrements to zero. The TXEOP bit in the SxINTE register can be set to enable an interrupt when an underrun event occurs.

For protocols other than USB and Ethernet, the EOP generator is bypassed.

5.6.3 SERDES Interrupts

Figure 5-7 shows the interrupt logic for the two SERDES units. For a detailed description of the SxINTE/SxINTF register bits, refer to Section 7.1.10. Note - If software sets a SerDes interrupt flag, the flag will be set again just after RETI, even if the flag is cleated in the ISR. Workaround is to disable GIE (Global Interrupt Enable is XCFG register bit 7) before the flag is set, then enable GIE after the flag is set..

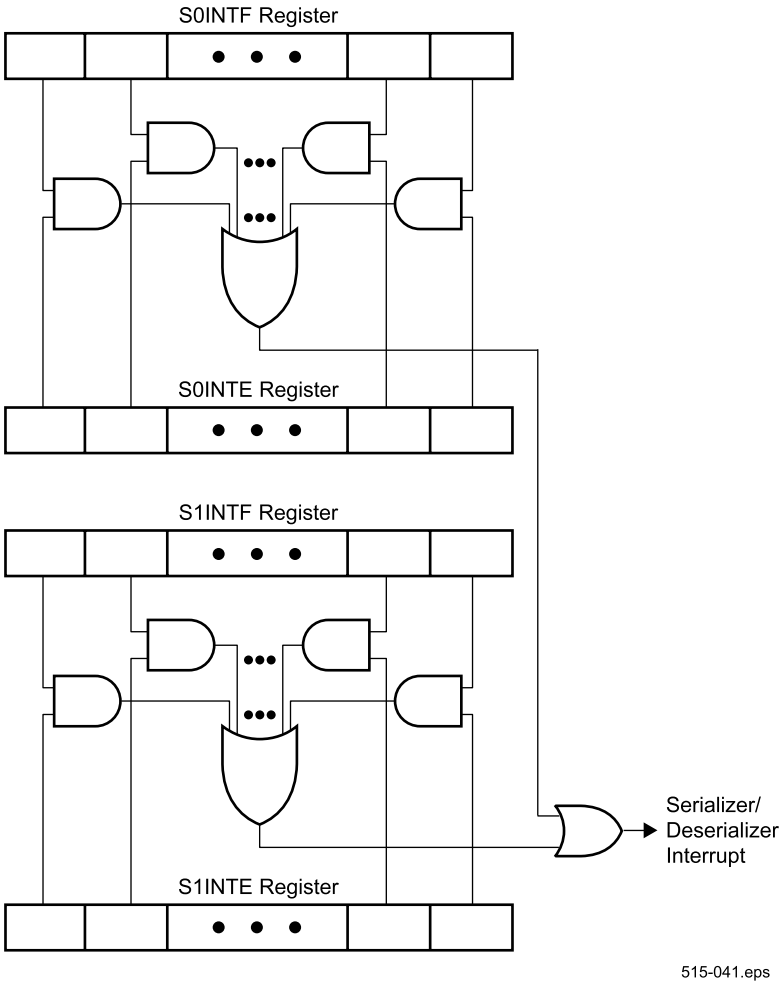


Figure 5-7 SERDES Interrupt Logic

5.6.4 Protocol Modes

Table 5-2 shows the features which are enabled for each protocol, as controlled by the PRS3:0 bits in the SxMODE register. These features affect which registers and register fields are used, for example the SxRSYNC register is only used in the USB and 10Base-T modes. The protocol mode also affects the signal usage, as shown in Table 5-3.

Table 5-2 Protocol Features

PRS3:0	Mode	Encoding Method	Differential or Single-Ended?	Synchronization Register Enabled?	EOP Generation/ Detection?	Bit Stuffing/ Unstuffing?	Pre-Emphasis Outputs Enabled?
0001	10Base-T	Manchester	Differential	Yes	Yes	N/A	Yes
0010	USB Bus	NRZI	Differential	Yes	Yes	Yes	N/A
0011	UART	None	Single-Ended	No	N/A	N/A	N/A
0101	SPI	None	Single-Ended	No	Yes	N/A	N/A
0110	GPSI	None	Single-Ended	No	Yes	N/A	N/A

Table 5-3 SERDES Protocol Modes And Pin Usage

Signal Names			SxCLK	SxRXP	SxRXM	SxRXD	SxTXPE	SxTXP	SxTXM	SxTXME	SxRX+	SxRX-
SERDES1 Pins			RE0	RE1	RE2	RE3	RE4	RE5	RE6	RE7	RG5	RG4
SERDES2 Pins			RF4	RF5	RF6	RF7	RF0	RF1	RF2	RF3	RG7	RG6
Mode	10Base-T Ethernet		-	-	-	RXD Note 1	TxD+ (O)	Tx+ (O)	Tx- (O)	TxD- (O)	RX+ (I)	RX- (I)
	USB Bus		Optional(I)	VP (I)	VM (I)	RCV (I)	OE (O)	VPO (O)	VMO (O)	-	-	-
	UART		Optional	-	-	RXD	-	TXD	-	-	-	-
	SPI	Master	SCLK	-	-	DI (I)	-	DO (O)	-	-	-	-
		Slave	SCLK	SS (I)	-	DI (I)	-	DO (O)	-	-	-	-
	GPSI	Master	-	TxEN (O)	-	TxD (O)	RxEN (I)	RxD (I)	TxCLK/ RxCLK (O)	TxBUSY (I)	-	-
		Slave	RxCLK (I)	TxEN (O)	-	TxD (O)	RxEN (I)	RxD (I)	TxCLK (I)	TxBUSY (I)	-	-

1. Used in comparator mode only.
I. Input
O. Output

Pins not used for protocols can be used for general I/O.

- *SxCLK* - Serial Clock in SPI or GPSI Slave modes, optional external SERDES clock input for USB or UART modes.
- *SxRXP* - Positive-side differential input (USB only), Slave Select (for SPI Slave), or data valid (GPSI).
- *SxRXM* - Negative-side differential input (USB only).
- *SxRXD* - Serial data for USB, UART, SPI and GPSI modes (10base-T Ethernet only when comparator is used).
- *SxTXPE* - Positive-side delayed differential output for pre-emphasis (10base-T Ethernet), output enable for external transceiver (USB), or data valid for GPSI mode.
- *SxTXP* - Positive-side differential output (10base-T Ethernet and USB modes), or serial data (UART, SPI and GPSI modes).
- *SxTXM* - Negative-side differential output (10base-T Ethernet and USB modes), transmit clock (GPSI Slave), or transmit and receive clock (GPSI Master).
- *SxTXME* - Negative-side delayed differential output for pre-emphasis (10base-T Ethernet), or TxBUSY in GPSI mode.
- *SxRX+* - Positive-side analog differential input, used for 10base-T Ethernet squelch function.
- *SxRX-* - Negative-side analog differential input, used for 10base-T Ethernet squelch function.

5.6.5 10base-T Ethernet

Hardware

Each SERDES unit provides 4 transmission digital signals: SxTXM, SxTXME, SxTXP, and SxTXPE. These signals for SERDES1 are multiplexed with the Port E pins, and the signals for SERDES2 are multiplexed with the Port F pins. The differential receive signals are multiplexed on port G pins for 10Base-T Ethernet mode: SxRX+ and SxRX-. The port direction bits must be set appropriately for each pin that is used. Other unused pins from SERDES and port G remain available for other functions usage. See Table 5-4 for details on signal port pin usage in various protocol modes.

Note: Proper operation of the 10Base-T requires that the core-clock be present as SERDES master clock is derived from it - don't turn off core clock while SERDES is still transmitting.

Figure 5-8 shows the clock/data separation and End-of-Packet (EOP) detection logic of a 10Base-T receiver unit. The SxRXP and SxRXM pins correspond to the differential inputs. Providing both inputs allows sensing of an EOP condition. To set up a SERDES unit for 10Base-T Ethernet, the input data from a differential line receiver is connected to the SxRX+ and SxRX- input. The signals designated Tx+, Tx-, TxD+, and TxD- correspond to the SxTXP, SxTXM, SxTXPE, and SxTXME pins of the corresponding SERDES. These pins are connected to an RJ45 jack through a transformer with terminations.

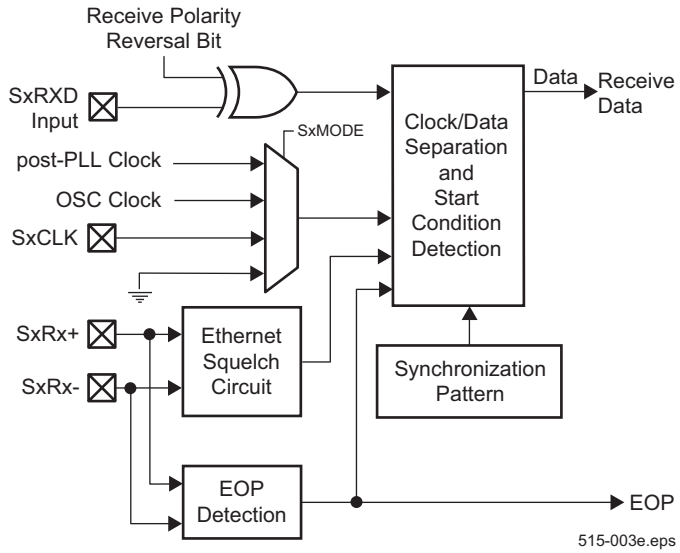


Figure 5-8 Clock/Data Separation and EOP Detection

Figure 5-9 shows an example circuit. RTXPE, RTXME, RTXP, RTXM and RL values vary depending on the Ethernet magnetics used. Please refer to IP2022 Native Ethernet application notes for more details.

For 10Base-T Ethernet operation, each SERDES is equipped with a squelch circuit for discriminating between noise, link pulses, and data. Link pulses are sent periodically to keep the channel open when no data is being transmitted. The squelch circuit handles link pulse detection, link pulse polarity detection, carrier sense, and EOP detection.

The 10Base-T mode requires only a fixed SFD (start of frame) pattern, so the SFD pattern for 10Base-T is hardwired to be 11010101 and the synchronization pattern register (SxRSYNC) is used to configure features of 10Base-T other than SFD pattern. Refer to Section 7.1.14 for detailed information. The incoming data stream, after passing through the polarity inversion logic (which can be turned on or off under software control) is compared to the synchronization pattern. Once a match is found, an internal counter is set to zero and data is shifted into a shift register. The synchronization matching operation is then disabled until an EOP condition is detected, because the synchronization pattern potentially could be embedded in the data stream as valid data.

Table 5-4 10base-T Ethernet Interface Signal and Port Pin Usage

10base-T Signal Name	SERDES Signal Name	SERDES1P in Name	SERDES2 Pin Name	Direction	Description
Tx+	SxTXP	RE5	RF1	Output	Plus-side differential output
Tx-	SxTXM	RE6	RF2	Output	Minus-side differential output
TxD+	SxTXPE	RE4	RF0	Output	Plus-side differential output with pre-emphasis
TxD-	SxTXME	RE7	RF3	Output	Minus-side differential output with pre-emphasis
Rx+	SxRX+	RG5	RG7	Input	Plus-side analog differential input, used for 10base-T Ethernet squelch function
Rx-	SxRX-	RG4	RG6	Input	Minus-side analog differential input, used for 10base-T Ethernet squelch function

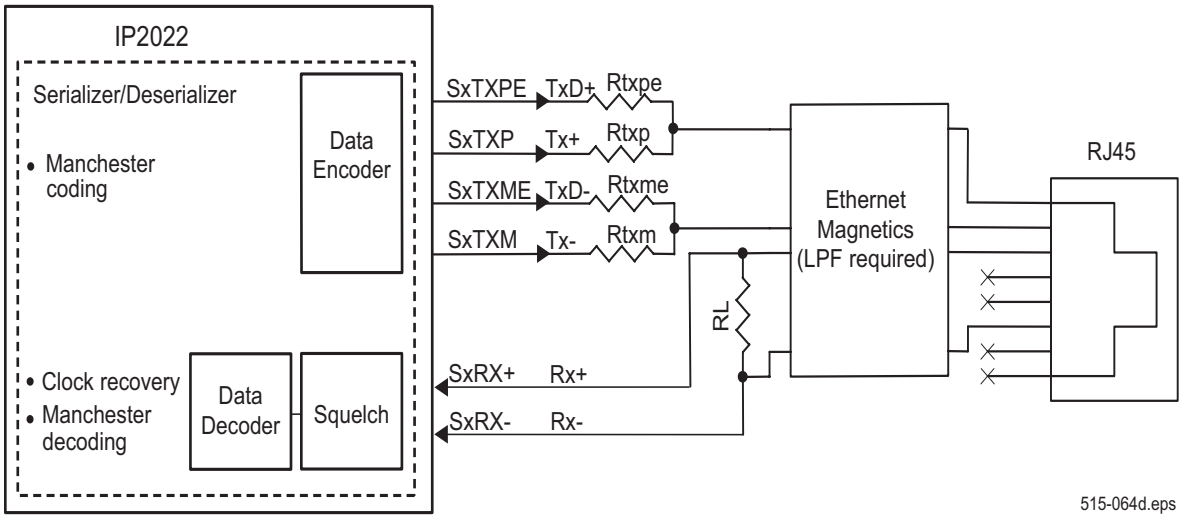


Figure 5-9 Ethernet Interface Example

Figure 5-10 shows the receive data paths. When an EOP is detected the SxRCNT register is loaded with the number of bits actually received, the EOP bit of the SxINTF register is set, and the data bits are loaded into the SxRBUF register. The RXBF bit in the SxINTE register can be set to enable an interrupt on this event.

The data encode block performs 10Base-T Manchester encoding. The encoded TX signal are sent to TX pins in a differential mode. The encode block is bypassed for all other protocols. The SxTXP and SxTXM pins have high current outputs for driving Ethernet magnetics directly without the use of transceivers. The pre-emphasis TX outputs are enabled on SxTXPE and SxTXME outputs, which have a 50ns-delayed and inverted version of the transmit outputs. The resistively combined TX outputs outside the chip are used to drive the magnetics. The output pins of the serializer are driven low when not transmitting.

Figure 5-11 shows the transmit data paths. For transmitting, software must specify the number of bits to transmit (specified in the SxTCFG register) and load the data in the SxTBUF register. This data is then transferred to an internal register, from which it is serially shifted out to the transmit logic. The TXBE bit in the SxINTE register can be set to enable an interrupt when the data has been transferred from the SxTBUF register. When there is a transmit buffer underrun event (i.e. all of the data has been shifted out from the internal register, but the SxTBUF register has not been reloaded), an EOP condition is automatically generated on the TX output pins after an internal counter decrements to zero. The TXEOP bit in the SxINTE register can be set to enable an interrupt when an underrun event occurs.

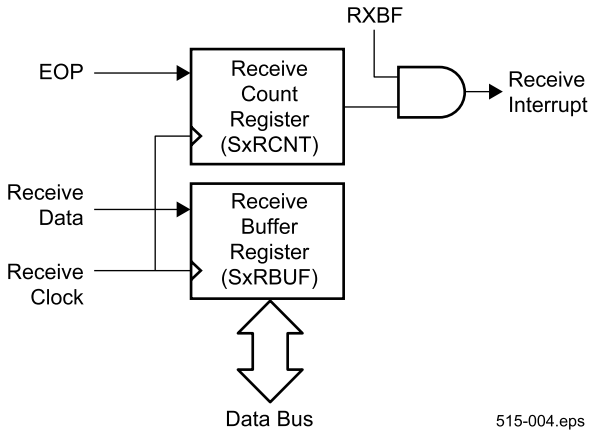


Figure 5-10 Receive Data Paths

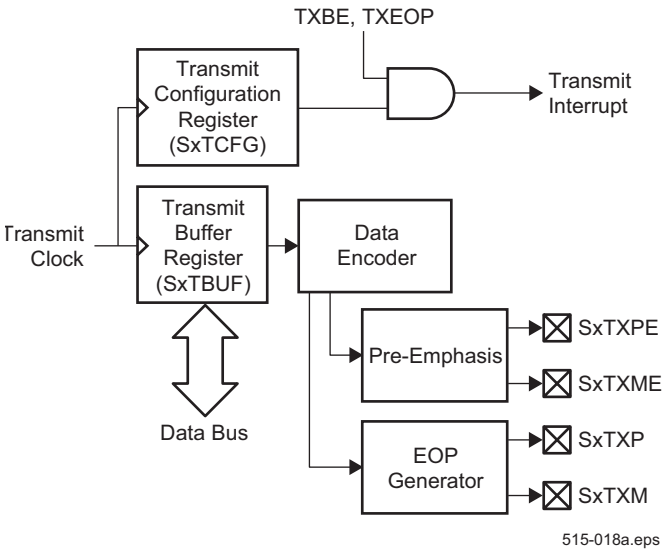


Figure 5-11 Transmit Data Paths

Software

The SERDES 10Base-T mode is designed to run at a fixed 8x oversampling for line receiver. So a fixed 80MHz (or multiple of 80MHz) master must be configured through SxMODE. The core PLL clock multiplier must be programmed to be an integer multiple of 80MHz for 10Base-T operation. The received data stream is used, together with the clock recovery circuit, to recover the original transmit clock and data.

Typical operation with Ubicom's SDK uses a 4.8 MHz crystal with a PLL post-divide by 2 to yield 120 MHz core operation.

Software must perform the following functions:

- Polarity detection and reversal.
- Carrier sense.
- Jabber detection.
- Link integrity test and link pulse generation.
- Random back off in case of collision.
- When a collision is detected, sending a 32-bit jam sequence. Collisions can be detected by positive detection of carrier sense during active transmission.
- Formation of Ethernet packet by putting the preamble, SFD, destination address, source address, length/type, MAC client data into the transmit buffer. Frame check computation can be done in software or through the LFSR units (see Section 5.9).
- MAC layer functions.

5.6.6 USB

Each SERDES provides support for USB revision 1.1 host and device modes of operation.

Hardware

To set up a SERDES unit for USB mode, the received data output of the USB transceiver should be connected to SxRXD. The VP and VM pins of the transceiver are connected to the SxRXP and SxRXM pins to allow detection of the EOP condition. Figure 5-12 shows the connections required between an external USB transceiver and the IP2012 / IP2022. Table 5-6 shows the mapping of USB signals to the SERDES pins. If desired, an external clock source can be connected to the SxCLK pin. For additional hardware configuration information, please refer to USB reference design available.

Software

The SxMODE register must be programmed with values for the desired USB mode, Full Speed or Low Speed. Serdes clock dividers, SxTMRH and SxTMRL, also need to be programmed to generate the appropriate frequency according to the USB submode selection. Table 5-7 shows the PLL clock frequencies required for the low and full speed modes of the USB mode. For example, if the PLL clock is 240 MHz, it can be programmed at 48 MHz for full speed with a divisor of 4 (=5). A divisor of 39 will make it suitable for low-speed operation. If external clock mode is selected, the clock divisor value is ignored and the clock is used directly for USB operation.

In USB mode, the SerDes uses two registers, SxRSYNC and SxSMASK, to detect the sync pattern marking the beginning of a USB data stream. This sequence is defined to be “7 zeros and a single 1” by the USB specification, and only the last 3 bits need to be matched to start receiving data, also defined in the specification. In order to achieve this, SxRSYNC needs to be programmed with 0x80 and SxSMASK needs to be programmed with 0xE0.

Receive behavior is controlled by the SxRCFG and SxRCNT registers. For USB operation, the higher 3 bits of SxRCFG should be set to zero, and the lower 5 bits should be set to the desired number of bits received, usually 8 or 16. SxRCNT should be cleared to make sure receive is performed LSB first. Once the SerDes matches the USB SYNC pattern, the internal receive count is reset to zero and the SerDes receives bits from the line until either the desired count is received or an EOP is encountered, at which point the received data is transferred to the Serdes Rx Data registers. If more data is coming in, the procedure will be repeated. Software is responsible for reading the data from the data registers before the next write by the hardware. Notice that this will be a short time, if RxCFG is configured with a small receive count, or an EOP is received before the desired count is reached. When the EOP is received, the SerDes remains idle until the next match of the SYNC pattern.

Transmit behavior is controlled by the SxTCFG register. For proper USB operation, the higher 3 bits should be set to 100, and the lower 5 bits should be set to the desired number of bits transmitted. Transmit can be initiated by writing to the SerDes Tx Data registers. Notice that both registers must be written even if the number of bits to transmit is less than 8. If the transmit count needs to be changed, it must be changed before the Tx Data registers are written. For continued transmission, Tx Data registers have to be written before the Tx bit count is reached. Otherwise, the SerDes automatically inserts the EOP signaling.

While receiving data, the clock/data separation circuit performs NRZI decoding, after which bit unstuffing is performed. This means every bit after a series of six consecutive ones is dropped. On transmit, the SerDes performs bit stuffing, and the clock/data separation circuit NRZI encodes the data.

Note: While configured for USB mode, the SerDes cannot be configured to interrupt on carrier status (RxCRS, SxRCNT bit 5; see Section 7.1.13).

Table 5-5 shows the function of SerDes status flags in USB Mode. See also Section 7.1.10.

Table 5-5 Summary of Status Flags in USB Mode

Flag	USB Mode Function
RXERROR	This bit indicates the presence of 7 ones in the USB bit stream.
RXEOP	This bit indicates the presence of EOP signaling, which is 2 bit times of SE0 and 1 bit time of J condition on the bus.
SYND	This bit indicates that SerDes successfully matched a USB SYNC pattern as configured in the SxSYNC register.
TXBE	This bit indicates that new data can be written to the Tx Data registers to continue uninterrupted data stream.
TXEOP	This bit indicates that an EOP condition was generated to signal the end of the USB data stream. Tx Data registers should be written with the sync pattern to start a new stream once this condition occurs.

Table 5-5 Summary of Status Flags in USB Mode

Flag	USB Mode Function
SXLINK-PULSE	TxIdle in USB mode. This bit indicates whether the SerDes is actively transmitting data.
RXBF	This bit indicates that Rx data registers are written with new data and should be read before the next desired amount of bits are received.
RXXCRS	This bit is a live status indicator in USB mode. If set, the SerDes is receiving data into the shift register, but there might not be any data in the data registers yet.

Software must perform the following functions to implement the USB protocol for a device:

- CRC generation and checking (can be done using the LFSR, Section 5.9).
- Detecting reset of the device function, which is indicated by 10 milliseconds of a single-ended zero (SE0) condition on the bus.
- Detecting the suspend state, which is indicated by more than 3 milliseconds of idle. Software muse make sure that the suspend current of 500 µA will be drawn after 10 milliseconds of bus inactivity.
- Formation of the USB packet by putting the sync, pid, and data into the transmit data registers and setting the proper count.
- Endpoint and device management and other higher level protocol tasks.

Timing Considerations

USB relies on certain timing limitations for error detection and recovery. Response time requirements are specifically harder to meet. ISR for USB needs to be carefully structured to satisfy these requirements, and this is possible because of IP2012 / IP2022's deterministic ISR execution times. The time from the SE0 on bus to the RXEOP indication is about 208 ns. The time from writing to TX data registers and the data put on the bus is about 125 ns. Software tasks like address, error, CRC checking, and determining the endpoint response have to be carefully timed and cycle counted to assure the required timing limitations are satisfied.

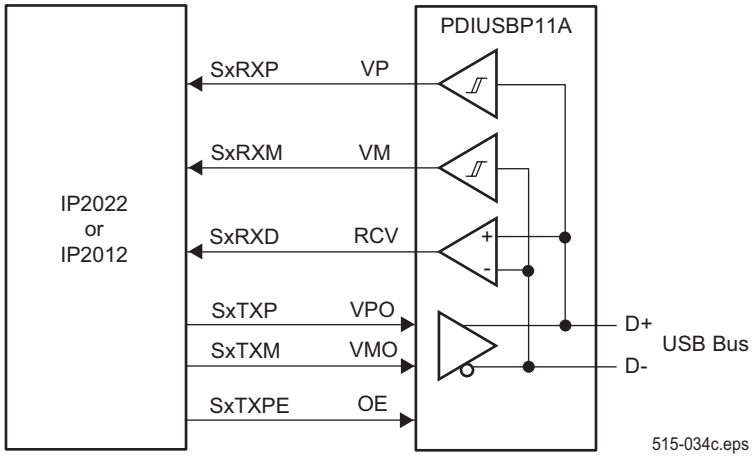


Figure 5-12 USB Interface Example

Table 5-6 USB Interface Signal Usage

USB Signal Name	SERDES Signal Name	SERDES1 Pin Name	SERDES2 Pin Name	Direction	Description
VP	SxRXP	RE1	RF5	Input	Plus-side differential input
VM	SxRXM	RE2	RF6	Input	Minus-side differential input
VPO	SxTXP	RE5	RF1	Output	Plus-side differential output
VMO	SxTXM	RE6	RF2	Output	Minus-side differential output

Table 5-6 USB Interface Signal Usage (continued)

USB Signal Name	SERDES Signal Name	SERDES1 Pin Name	SERDES2 Pin Name	Direction	Description
OE	SxTXPE	RE4	RF0	Output	Output enable
RCV	SxRXD	RE3	RF7	Input	Receive data
Clock	SxCLK	RE0	RF4	Input	External clock input (optional)

Table 5-7 Required Clock Frequencies from SerDes Clock in USB Mode

Protocol	Receive
USB 1.1 Full Speed	48 MHz
USB 1.1 Low Speed	6 MHz

5.6.7 UART

For UART operation, two internal divide-by-16 circuits are used. Based on the clock source (either internal or external), the receive section and the transmit section use two divided-by-16 clocks that potentially can be out of phase. This is due to the nature of the UART bus transfers. The receive logic, based on the 16x bit clock (the clock source chosen by user), will sample the incoming data for an falling edge. Once the edge is detected, the receive logic counts 8 clock cycles and samples the number of bits specified in the SxRCNT register using the bit clock (which is obtained by dividing the clock source by 16).

Hardware

Figure 5-13 shows an example circuit to connect the SERDES in UART mode. Table 5-8 shows the UART signal to port pin usage.

Software

To set up a SERDES unit for UART mode, select UART mode in the PRS3:0 bits of the SxMODE register. This causes the data to be clocked in after a valid start bit is detected. Make sure that the polarity selected by the RPOREV bit in the SxRCFG register and the TPOREV bit in the SxTCFG register match the polarity provided by the RS-232 transceiver. (Most of them are inverted.) Make sure the bit order is compatible with the data format (RS-232 uses LSB-first bit order). The receiver uses 16X oversampling, so select a SERDES clock divisor (see Section 7.1.17 for information on the SxTMRH/L registers) that is 16 times the desired baud rate.

To operate in UART mode, depending on the application, either transmit or receive can be performed first. In both cases, the configuration register needs to be programmed with a bit count that is appropriate for the format. The bit count depends on the number of data bits, stop bits, and parity bits. The start bit is included in the bit count. The receiver does not check for the presence of stop bits. To detect framing errors caused by missing stop bits, increase the receiver's bit count (i.e. the RXSCNT field in the SxRCFG register) and test the trailing bit(s) in software.

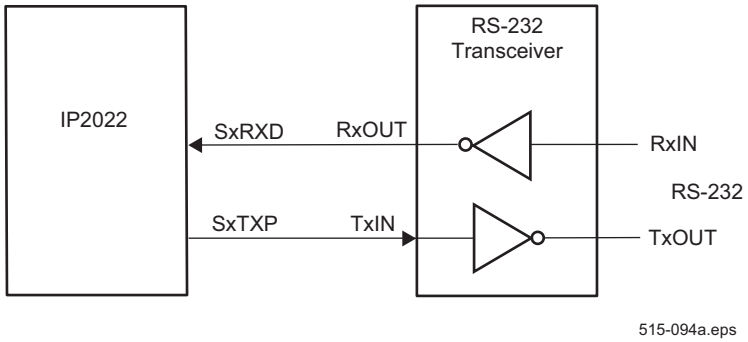


Figure 5-13 UART Interface Example

Table 5-8 UART Interface Signal Usage

UART Signal Name	SERDES Signal Name	SERDES1 Pin Name	SERDES2 Pin Name	Direction	Description
RXD	SxRXD	RE3	RF7	Input	Receive data
TXD	SxTXP	RE5	RF1	Output	Transmit data

5.6.8 SPI

Hardware

Figure 5-14 shows example circuits to connect the SERDES in SPI mode. Table 5-10 and Table 5-11 show the SPI signal to port pin usage.

Configuration

The SERDES can be configured for either master or slave mode:

SxRCFG[7] = 0: slave
SxRCFG[7] = 1: master

The SERDES SCK idle-level (i.e. when $\overline{SS}$ is de-asserted) can be configured:

SxMode[3] (CPOL) = 0: idle is low
SxMode[3] (CPOL) = 1: idle is high

Finally, the SERDES can be configured for the phase relationship of the SDO/SDI pins with respect to the SCK edge:

SxMode[2] (CPHA) = 0:
SDO is set up by the other device a half clock period before the first edge following the assertion of SS*. Therefore SDI will be sampled by this device (the slave) on the first edge (transition).

SxMode[2] (CPHA) = 1:
SDO is set up by the other device on the first edge following the assertion of SS*. Therefore SDI will be sampled by this device (the slave) on the second edge (transition).

Note: The use of the term “edge” in the above paragraphs implies any transition, not a specific type (i.e. rising or falling) of transition. Therefore, “first edge” implies a rising edge when CPOL=0, and implies a falling edge when CPOL=1.

In the SPI scheme implemented by Motorola, which the IP2012 / IP2022 follows, data being output on SDO and data being sampled on SDI always occur on opposing edges of the clock, on either master or slave. Transmitting and sampling on the same-edge of the clock is not supported by the SERDES.

CPOL, in conjunction with CPHA, determines which clock edges the SERDES will be using to output and sample data on, as given by Table 5-9.

Table 5-9 SERDES Output and Sample Configuration

CPOL	CPHA	
0	0	output on falling, sample on rising
0	1	output on rising, sample on falling
1	0	output on rising, sample on falling
1	1	output on falling, sample on rising

When the SERDES is configured as a slave, the state of the SDO line when SS* is de-asserted will be determined by the value in the RxOUT GPIO register for that pin, which the user can configure.

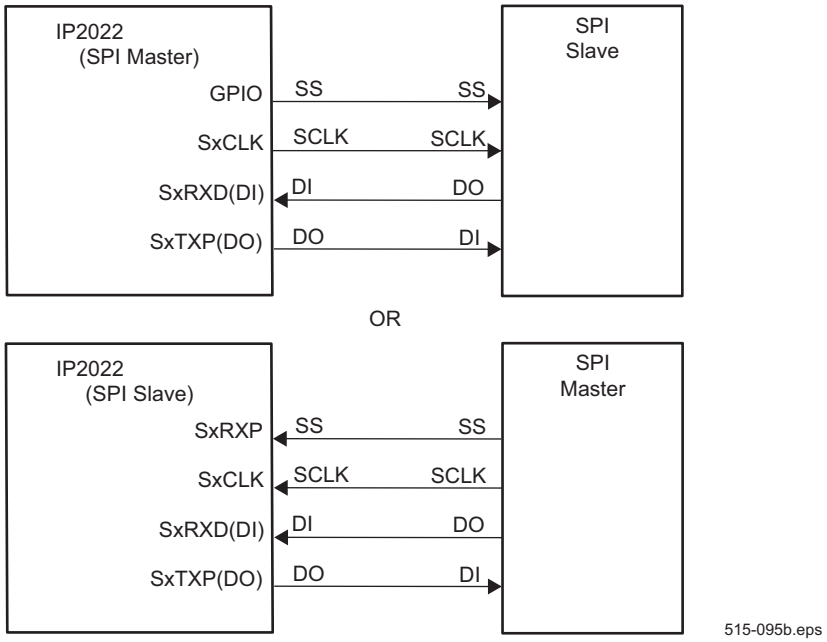


Figure 5-14 SPI Interface Examples

Table 5-10 SPI Master Interface Signal Usage

SPI Device Signal Name	IP2022 SPI Signal Name	SERDES Signal Name	SERDES1 Pin Name	SERDES2 Pin Name	Direction	Description
SCLK	SCLK	SxCLK	RE0	RF4	Output	Serial clock output in master mode, input in slave mode
DO	DI	SxRXD	RE3	RF7	Input	Receive data
DI	DO	SxTXP	RE5	RF1	Output	Transmit data
SS	SS	GPIO	RE1	RF5	Output	Slave select pin used in slave mode only (Master select handled by software)

Table 5-11 SPI Slave Signal Usage

SPI Device Signal Name	IP2012 / IP2022 SPI SIgnal Name	SERDES Signal Name	SERDES1 Pin Name	SERDES2 Pin Name	Direction	Description
SCLK	SCLK	SxCLK	RE0	RF4	Input	Serial clock output in master mode, input in slave mode
DO	DI	SxRXD	RE3	RF7	Input	Receive data
DI	DO	SxTXP	RE5	RF1	Output	Transmit data
SS	SS	SxRXP	RE1	RF5	Input	Slave select pin used in slave mode only (Master select handled by software)

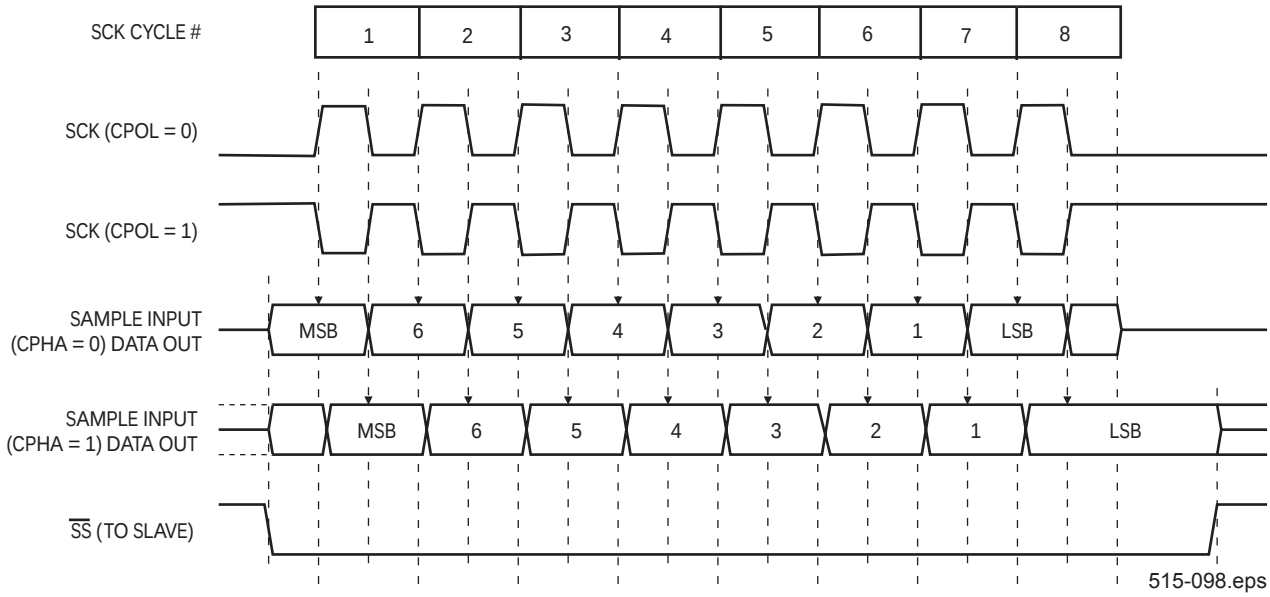


Figure 5-15 SPI Signal Timing

5.6.9 GPSI

Hardware

Figure 5-16 shows example circuits to connect the SERDES in GPSI (General Purpose Serial Interface) mode. Table 5-12 shows the GPSI signal to port pin mapping in Master mode, and Table 5-13 shows the GPSI signal to port pin mapping in Slave mode.

Software

GPSI is a general-purpose, point-to-point, full-duplex serial bus protocol. Only two devices are allowed to exist on a bus. The GPSI PHY device is responsible for maintaining bus timing by driving two continuously running clocks, TxClk and RxClk. The device that does not drive the clocks is the MAC device. The TxEn and TxD signals are synchronized to the TxClk clock. The RxD and RxEn signals are synchronized to the RxClk clock.

The COLLISION and TxBUSY signals do not participate in actual data transfer on the GPSI bus. COLLISION and TxBUSY provide additional flow control capabilities for the software device driver. The COLLISION signal is used to indicate that a PHY device has detected a collision condition. This signal is only useful when the SERDES is connected to a PHY device or acting as a PHY device.

The TxBUSY signal is used by a GPSI device to indicate that the device is currently busy, and that another device should not attempt to start a data transfer.

Refer to SxRCNT register bit 5. Refer to Section 7.1.10 through Section 7.1.17 for detailed configurations.

Example: To get the PLL clock divided by 2 out on RE6:

```
S1MODE=63
S1RCFG=80
S1TCFG=80
```

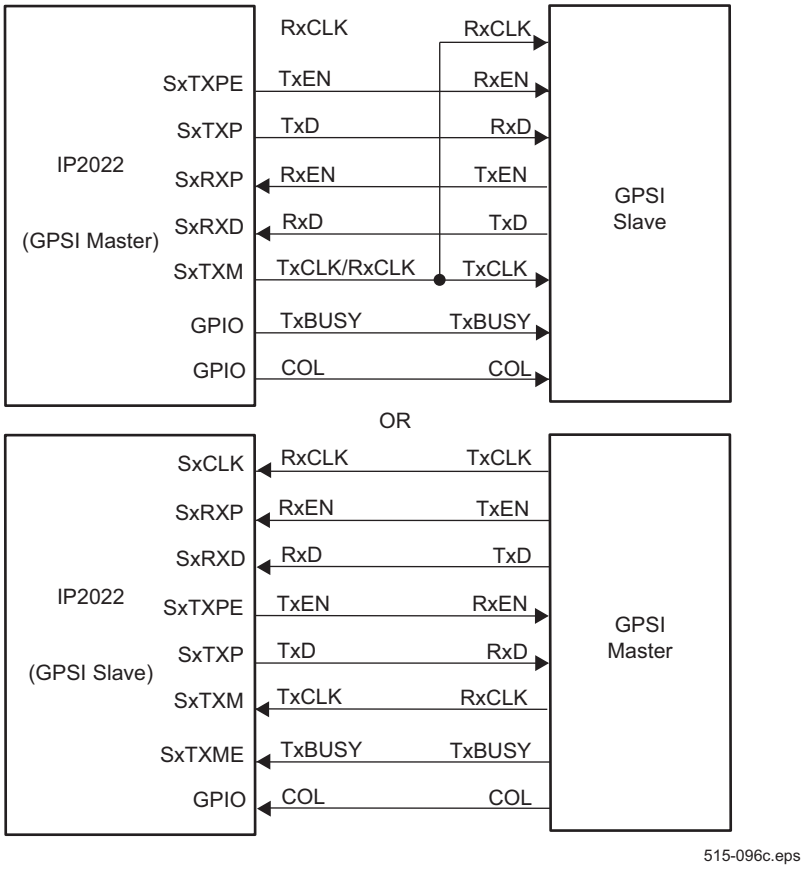


Figure 5-16 GPSI Interface Examples

Table 5-12 GPSI Master Interface Signal Usage

GPSI Slave Signal Name	IP2022 GPSI Signal Name	SERDES Signal Name	SERDES1 Pin Name	SERDES2 Pin Name	IP2022's Direction	Description
TxCLK and RxCLK	TxCLK and RxCLK	SxTXM	RE6	RF2	Output	Transmit and Receive clock
TxD	RxD	SxRXD	RE3	RF7	Input	Transmit data
TxEN	RxEN	SxRXP	RE1	RF5	Input	Transmit data valid
RxD	TxD	SxTXP	RE5	RF1	Output	Receive data
RxEN	TxEN	SxTXPE	RE4	RF0	Output	Receive data valid
TxBUSY	TxBUSY	GPIO	-	-	Output	Indicates a data transfer in progress (handled by software)
COLLISION	COLLISION	GPIO	-	-	Output	Indicates a collision at PHY layer (handled by software)

Table 5-13 GPSI Slave Interface Signal Usage

GPSI Master Signal Name	IP2022 GPSI Signal Name	SERDES Signal Name	SERDES1 Pin Name	SERDES2 Pin Name	IP2022's Direction	Description
RxCLK	TxCLK	SxTXM	RE6	RF2	Input	Transmit clock
RxD	TxD	SxTXP	RE5	RF1	Output	Transmit data
RxEN	TxEN	SxTXPE	RE4	RF0	Output	Transmit data valid
TxCLK	RxCLK	SxCLK	RE0	RF4	Input	Receive clock
TxD	RxD	SxRXD	RE3	RF7	Input	Receive data
TxEN	RxEN	SxRXP	RE1	RF5	Input	Receive data valid
TxBUSY	TxBUSY	SxTXME	RE7	RF3	Input	Indicates a data transfer in progress (handled by software)
COLLISION	COLLISION	GPIO	-	-	Input	Indicates a collision at PHY layer (handled by software)

Note: In GPSI master mode, the SxTXM SERDES pin should be used by the GPSI slave for both TxCLK and RxCLK inputs.

5.7 Analog to Digital Converter (ADC)

The on-chip A/D converter has the following features:

- 10-bit ADC (when Vref > 2.3V)
- 8 input channels
- 48 kHz maximum sampling rate (min 21us)
- One-shot conversion.
- Optional external reference voltage
- Vmax = AVdd (max 2.7V)
- Result returned in the ADCH and ADCL registers

Figure 5-17 shows the A/D converter circuitry. The ADC input pins use alternate functions of the Port G pins. The result of an ADC sample is the analog value measured on the selected pin. To correctly read an external voltage, the pin being sampled must be configured as an input in the port direction register (i.e. the RGDIR register). If the pin is configured as an output, then the result will indicate the voltage level being driven by the output buffer. The RG1 and RG2 port pins are also used as the analog comparator input pins. The result of sampling the RG1 or RG2 pins will be correct whether or not the comparator is operating. The RG0 pin is also used as the comparator output pin. If the comparator is enabled, then sampling the RG0 pin will indicate the voltage level being driven by the comparator. The RG3 pin is multiplexed with the external reference voltage.

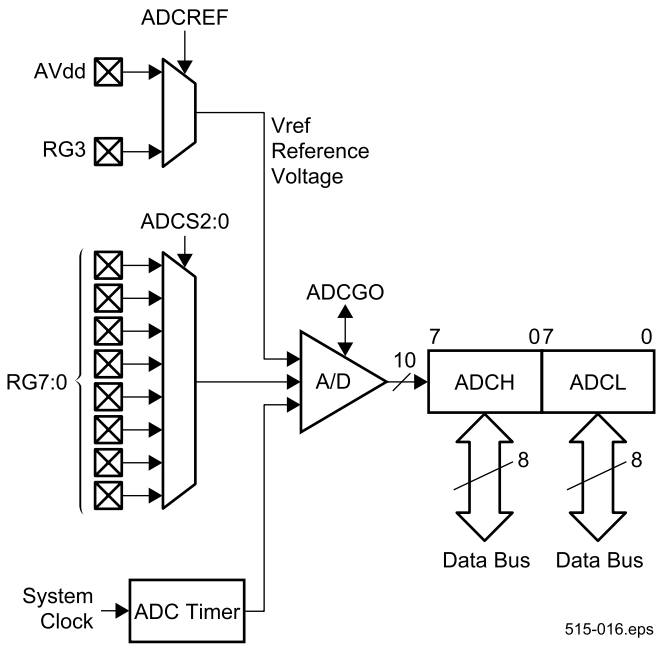


Figure 5-17 A/D Converter Block Diagram

5.7.1 ADC Reference Voltage

The reference voltage (Vref) can come from either the RG3 port pin or from the AVdd supply voltage. If AVdd is used, the RG3 port pin may be used as a channel of analog input or as a general-purpose port pin.

Vref defines a voltage level which reads as one increment of resolution below the full-scale voltage. The full-scale voltage reads as 0x3FF, so the Vref voltage reads as 0x3FE and the A/D converter resolution is 10 bits. Table 5-14 shows the values reported at the upper and lower limits of the ADC input voltage range.

5.7.2 A/D Converter Registers

ADCTMR Register

The ADCTMR register is used to specify the number of system clock cycles required for a delay of at least 868ns. The ADCTMR times out 24 times before the ADC circuitry stops its measurement. This gives the proper sampling rate of 24 x 868ns = 20.83us (48kHz).

To calculate the ADCTMR decimal value that should be used, use ADCTMR = (system clock in MHz)/1.152.

For example, at a system clock frequency of 120 MHz, the timer register should be set to 120/1.152 = 104 decimal or 68 hex. The minimum value that may be loaded into the ADCTMR register is 2, so the system clock must be at least 24 times the ADC sampling frequency for the ADC to function (system clock must be at least 48kHz x 24 = 1.152MHz).

ADCCFG Register

The A/D converter configuration register (ADCCFG) provides the control and status bits for the A/D converter, as shown in Section 7.1.1.

Table 5-14 ADC Values

Vin Voltage	ADC Value
0V	0x000
Vref/0x3FE	0x001
Vref	0x3FE
Vref + (Vref/0x3FE)	0x3FF

5.7.3 Using the A/D Converter

The following sequence is recommended:

1. Set the ADCTMR register to the correct value for the system clock speed.
2. Load the ADCCFG register to specify the channel and set the ADCGO bit. Setting the ADCGO bit enables and resets the ADC timer.
3. After a period of time (24 timer overflows = 20.8 µs) the conversion will complete, the ADCGO bit will be cleared, and the ADC timer will be disabled.
4. A timer-based interrupt service routine can detect or assume the ADCGO bit has been cleared and read the ADC value.
5. Another load to the ADCCFG register can then be used to start another conversion.

5.7.4 ADC Result Justification

The 10 bits of the ADC value can be mapped to the 16 bits of the ADCH/ADCL register pair in three different ways, as shown in Table 5-15. In this table, the numbers in the cells represent bit positions in the 10-bit ADC value, Z represents zero (as opposed to bit position 0), and -9 represents the inversion of bit position 9.

Table 5-15 Justification of the ADC Value

Mode	ADCH Register Bits								ADCL Register Bits							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Left Justified	9	8	7	6	5	4	3	2	1	0	Z	Z	Z	Z	Z	Z
Right Justified	Z	Z	Z	Z	Z	Z	9	8	7	6	5	4	3	2	1	0
Signed	-9	-9	-9	-9	-9	-9	-9	8	7	6	5	4	3	2	1	0

5.8 Comparator

The IP2012 / IP2022 has an on-chip analog comparator which uses alternate functions of the RG0, RG1, and RG2 port pins. The RG1 and RG2 pins are the comparator negative and positive inputs, respectively, while the RG0 pin is the comparator output pin. To use the comparator, software must program the port direction register (RGDIR) so that RG1 and RG2 are inputs. RG0 may be set up as a comparator output pin.

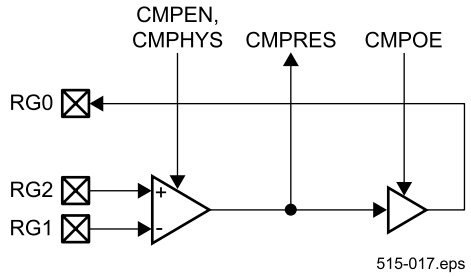


Figure 5-18 Analog Comparator

The comparator enable bits are cleared on reset, which disables the comparator. To avoid drawing additional current during power-down mode, the comparator should be disabled before entering power-down mode. A 50 mV hysteresis is applied between the inputs, when the CMPHYS bit is set in the CMPCFG register.

5.8.1 CMPCFG Register

The CMPCFG register is used to enable the comparator, to read the output of the comparator internally, to enable the output of the comparator to the comparator output pin, and to enable the hysteresis. Section 7.1.3 shows the bits in this register.

5.9 Linear Feedback Shift Register (LFSR)

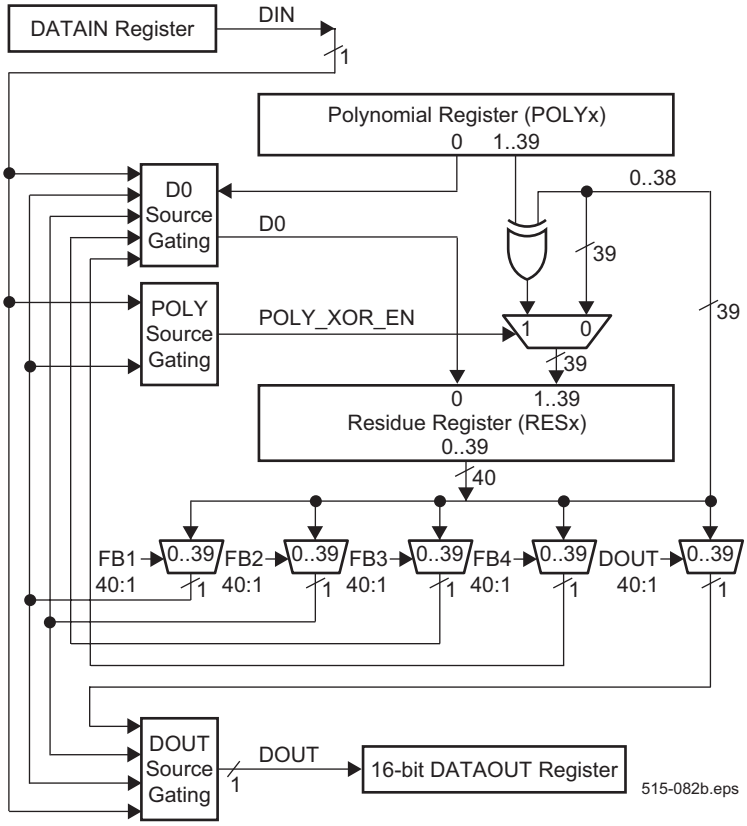


Figure 5-19 LFSR Block Diagram

Four linear feedback shift register (LFSR) units provide hardware support for the computation-intensive inner loops of algorithms commonly used in data communications, such as:

- Cyclic Redundancy Check (CRC)
- Data Scrambling
- Data Whitening
- Encryption/Decryption
- Hashing

The LFSR units implement a programmable architecture, which can be adapted for algorithms used by the Bluetooth, Ethernet, Homeplug, HomePNA, HomeRF, IEEE 802.11, and USB communication protocols. Figure 5-19 is a block diagram of the LFSR architecture.

The 40-bit residue register and its surrounding circuits are the computational core of an LFSR unit. On every clock cycle, 39 output bits from the register are available at the input for performing a shift operation or a polynomial add/subtract-and-shift operation. Four 40-bit multiplexers at the output of the residue register allow selecting up to four terms of the register for feedback into the input (D0), polynomial operation control (POLY_XOR_EN), and output (DOUT) bit streams. A fifth multiplexer is only used for generating the output bit stream.

The polynomial and residue registers are mapped as five 8-bit registers. The mapping of the residue register is controlled by the ML_OUT bit of the LFSRCFG3 register, as shown in Figure 5-20.

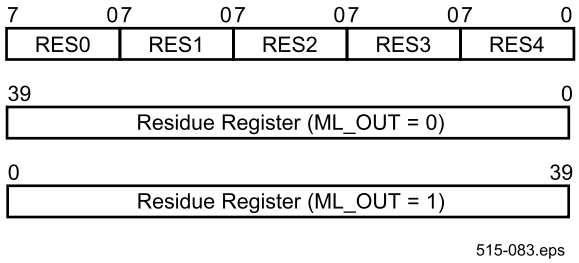


Figure 5-20 Mapping of the Residue Register

Input data is shifted serially out of the 16-bit DATAIN register, which can be programmed to provide the data LSB-first or MSB-first. Output data is shifted serially LSB-first into the 16-bit DATAOUT register.

A 32-bit RESCMP register (not shown) can be used to compare the result in the residue register against an expected value. When ML_OUT is set, residue register bits 0:31 are compared against RESCMP bits 0:31. When ML_OUT is clear, residue register bits 39:8 are compared to RESCMP bits 0:31, respectively. If there are bits in the residue register which do not participate in the programmed LFSR operation, be sure that the corresponding bits in the RESCMP register are initialized to the same values as these non-participating bits.

Each LFSR unit has three configuration registers (LFSRCFG1, LFSRCFG2, and LFSRCFG3) for various control and status bits. The HL_TRIGGER bit in the LFSRCFG3 register controls whether operation of the LFSR unit is triggered by a write to the high byte or the low byte of the DATAIN register (i.e. DATAINH or DATAINL). The operation then proceeds for some number of cycles programmed in the SHIFT_COUNT3:0 field of the LFSRCFG1 register. Completion of the operation is indicated when the DONE bit in the LFSRCFG1 register is set. (Alternatively, software can wait 1 cycle/bit of DATAIN before reading the result.)

An autoloading option is available for each LFSR unit to load the DATAIN register automatically from the SERDES RX buffers (SxRBUF register of the corresponding SERDES unit). LFSR0 and LFSR2 are paired with SERDES1, and LFSR1 and LFSR3 are paired with SERDES2.

Three registers in data memory are used to access the LFSR register banks, as shown in Table 5-16.

Table 5-16 LFSR Registers in Data Memory

Address	Name	Description
0x23	LFSRH	High data byte
0x27	LFSRL	Low data byte
0x2B	LFSRA	Address register

The LFSRA register is loaded to point to a specific LFSR unit and a register pair within the unit. The LFSRA register has the format shown in Figure 5-21

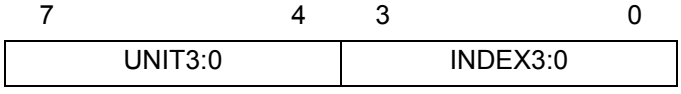


Figure 5-21 LFSRA Register

Only 0, 1, 2, and 3 are valid as the UNIT, LFSRA bits 7 and 6 = don't care. The valid encodings for the index are shown in Table 5-17.

Table 5-17 LFSRA Register INDEX Encoding

INDEX3:0	High Byte (LFSRH)	Low Byte (LFSRL)
0x0	DATAINH	DATAINL
0x1	DATAOUTH	DATAOUTL
0x2	FB2	FB1
0x3	LFSRCFG2	RES4
0x4	RES3	RES2
0x5	RES1	RES0
0x6	FB4	FB3
0x7	LFSRCFG3	DOUT
0x8	LFSRCFG1	POLY4
0x9	POLY3	POLY2
0xA	POLY1	POLY0
0xB	RESCMP3	RESCMP2
0xC	RESCMP1	RESCMP0
All registers initialized to 0x00 upon reset, except RESx = 0xFF, RESCMPx = 0xFF, and LFSRCFG1 = 0x10).		

The LFSR registers do not support consecutive read-modify-write operations. For example, the following instruction sequence loads unpredictable values:

```
clrb lfsrh,7
clrb lfsrh,4
```

5.9.1 LFSRCFG1 Register

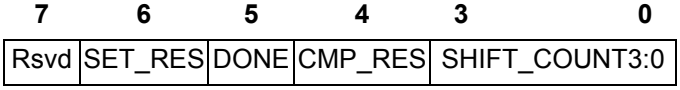


Figure 5-22 LFSRCFG1 Register

- *SET_RES*—set to initialize the residue register to all ones (write-only, reads as zero).
- *DONE*—clear while the LFSR is busy, set when the operation is completed (read only).
- *CMP_RES*—set if last LFSR operation result matched contents of RESCMP register (read only).
- *SHIFT_COUNT3:0*—specifies number of bits to shift, load with N for an operation of N+1 shifts.

5.9.2 LFSRCFG2 Register

7	6	5	4	3	2	1	0
DOUT_DOUT_EN	DIN_DOUT_EN	FB1_DOUT_EN	FB2_DOUT_EN	DIN_D0_EN	FB1_D0_EN	FB2_D0_EN	DATA_IN_POLYXOR_EN

Figure 5-23 LFSRCFG2 Register

- *DOUT_DOUT_EN*—set to enable DOUT multiplexer output in source gating for DOUT node.
- *DIN_DOUT_EN*—set to enable DIN signal in source gating for DOUT node.
- *FB1_DOUT_EN*—set to enable FB1 signal in source gating for DOUT node.
- *FB2_DOUT_EN*—set to enable FB2 signal in source gating for DOUT node.
- *DIN_D0_EN*—set to enable DIN signal in source gating for D0 node.
- *FB1_D0_EN*—set to enable FB1 signal in source gating for D0 node.
- *FB2_D0_EN*—set to enable FB2 signal in source gating for D0 node.
- *DATA_IN_POLYXOR_EN*—set to enable DIN signal in source gating for POLY_XOR_EN node.

5.9.3 LFSRCFG3 Register

7	6	5	4	3	2	1	0
Reserved		AUTOLOAD_EN	ML_OUT	ML_IN	HL_TRIGGER	FB3_D0_EN	FB4_D0_EN

Figure 5-24 LFSRCFG3 Register

- *AUTOLOAD_EN*—set to enable autoloading DATAIN register when SxRBUF register of corresponding SERDES unit is loaded.
- *ML_OUT*—set to shift data out of residue register LSB, and into MSB, clear to shift data out of residue register MSB, and into LSB. See Figure 5-20 for effect on RESx mapping.
- *ML_IN*—set to shift data from DATAIN register MSB-first to DIN node, clear to shift data LSB-first.
- *HL_TRIGGER*—set to trigger operation start on loading DATAINH register, clear to trigger on DATAINL.
- *FB3_D0_EN*—set to enable FB3 signal in source gating for D0 node.
- *FB4_D0_EN*—set to enable FB4 signal in source gating for D0 node.

5.9.4 DATAIN Register

The 8-bit DATAINH and DATAINL registers together comprise the 16-bit DATAIN register. For LFSR0 and LFSR2, the AUTOLOAD_EN bit in the LFSRCFG3 register can be used to enable automatic loading from SERDES1. For LFSR1 and LFSR3, the AUTOLOAD_EN bit in the LFSRCFG3 register can be used to enable automatic loading from SERDES2. The HL_TRIGGER bit in the LFSRCFG3 register controls whether loading the DATAINH or DATAINL register triggers the start of the LFSR operation. The ML_IN bit in the LFSRCFG3 register controls whether data is shifted MSB-first or LSB-first from the DATAIN register to the DIN node.

5.9.5 DATAOUT Register

The 8-bit DATAOUTH and DATAOUTL registers together comprise the 16-bit DATAOUT register. Data shifted out of the residue register is shifted LSB-first into the DATAOUT register.

5.9.6 DOUT Register

The DOUT register controls a 40:1 multiplexer on the residue register outputs. It selects a term which can be used in the source gating for the DOUT bit stream.

5.9.7 FBx Registers

The four FBx registers control four 40:1 multiplexers on the residue register outputs. They select feedback terms which can be used in the source gating for the D0, POLY_XOR_EN, and DOUT bit streams.

5.9.8 POLYx Registers

The five POLYx registers hold the 40-bit polynomial used in the LFSR operation.

5.9.9 RESx Registers

The five RESx registers hold the 40-bit residue used in the LFSR operation. The ML_OUT bit controls the mapping of

the residue register to the RESx registers, as shown in Figure 5-20. The residue register can be initialized to all ones by setting the SET_RES bit in the LFSRCFG1 register.

5.9.10 RESCMPx Registers

The four RESCMPx registers hold a 32-bit value for comparison with the contents of the residue register. After an LFSR operation is completed, the CMP_RES bit in the LFSRCFG1 register indicates whether the result of the operation matched the 32-bit value. When the ML_OUT bit in the LFSRCFG3 register is clear, bits 39:8 of the residue register are compared against bits 0:31 of the RESCMP register. When ML_OUT is set, bits 0:31 of the residue register are compared against bits 0:31 of RESCMP.

5.9.11 LFSR Configuration

The LFSR units were designed with the following communication protocols in mind: Bluetooth, Ethernet, Homeplug, HomePNA, HomeRF, IEEE 802.11, and USB. Table 5-18 shows the LFSR configurations used to support these protocols.

Table 5-18 LFSR Configurations for Various Protocols

Protocol	Subfunction	D0 In	Feedback	D Out
USB	CRC16	Din^D15	Din^D15	
	CRC5	Din^D4	Din^D4	
Ethernet	CRC32	Din^D31	Din^D31	
	Scrambler	Din^D17^D22		Din^D17^D22
	Descrambler	Din		Din^D17^D22
HomePlug	CRC8	Din^D7	Din^D7	
	CRC16	Din^D15	Din^D15	
	Scrambler	D6^D3		Din^D6^D3
HomePNA	CRC8	Din^D7	Din^D7	
	CRC16	Din^D15	Din^D15	
	Scrambler	D17^D22		Din^D17^D22
802.11	CRC32 (FCS)	Din^D31	Din^D31	
	CRC16 (HEC)	Din^D15	Din^D15	
	Data Whitening	D3^D6		Din^D3^D6
	CRC16 (CRC)	Din^D15	Din^D15	
	Scrambler	Din^D3^D6		Din^D3^D6
	Descrambler	Din		Din^D3^D6
Home-RF	CRC	Din^D31	Din^D31	
	Scrambler	Din		Din^D3^D8
	Descrambler	Din		Din^D3^D8
Bluetooth	FEC	Din^D4	Din^D4	
	HEC	Din^D7	Din^D7	
	CRC16	Din^D15	Din^D15	
	Data Whitening	D6	D6	Din^D6
	Encryption	Din^D8^D12^D20^D25		D24
		Din^D12^D16^D24^D31		D24
		Din^D4^D24^D28^D33		D32
		Din^D4^D28^D36^D39		D32

5.10 Parallel Slave Peripheral (PSP)

The Parallel Slave Peripheral allows the IP2022 to operate as an 8- or 16-bit slave to an external device, much like a memory chip. The IP2022 supports either 8-bit or 16-bit wide bus operation. Alternate functions of Port C and Port D are used for transferring data, and alternate functions of Port B are used for control signals. Figure 5-25 shows the connections between an external master and the Parallel Slave Peripheral interface.

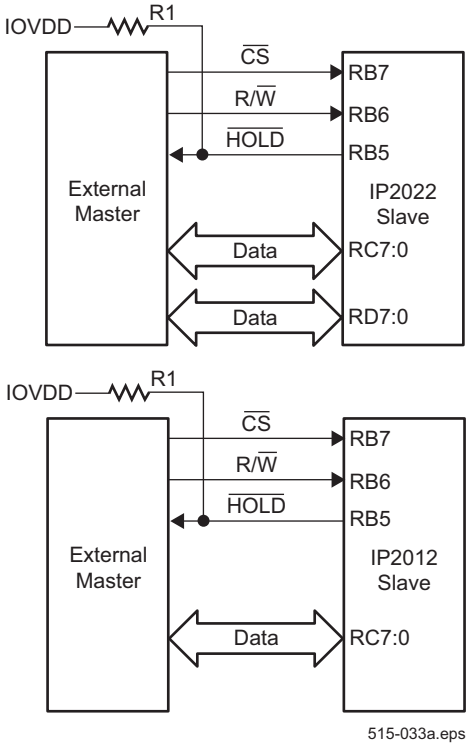


Figure 5-25 Parallel Slave Peripheral

To read or write through the Parallel Slave Peripheral interface, the external master asserts the chip select ($\overline{CS}$) signal low. This signal is an alternate function of port pin RB7. The direction of transfer is indicated by the R/$\overline{W}$ signal, which is an alternate function of port pin RB6. When the R/$\overline{W}$ signal is high, the master is reading from the slave. When the R/$\overline{W}$ signal is low, the master is writing to the slave.

Optionally, a $\overline{HOLD}$ signal may be enabled as an alternate function of port pin RB5. Assertion of $\overline{HOLD}$ indicates to the external master that the Parallel Slave Peripheral interface is not ready to allow the data transfer to complete. The $\overline{HOLD}$ signal is driven like an open-collector signal, i.e. low when asserted and high-impedance when not asserted. When PSP is enabled and the $\overline{CS}$ signal is high (i.e. the IP2012 / IP2022 is not selected), the $\overline{HOLD}$ signal is in high-impedance mode, and the C and D ports are tri-stated. The $\overline{HOLD}$ signal should have an external pullup resistor (R1 = 10K Ω is recommended). The $\overline{CS}$ signal must not be allowed to float.

When $\overline{CS}$ is asserted, an interrupt is generated and $\overline{HOLD}$ (if enabled) is automatically asserted. If the data transfer is a write from the external master, software reads the Port C, Port D, or both. If the data transfer is a read, software writes the data to the port or ports. Finally, if $\overline{HOLD}$ is asserted, software releases assertion of $\overline{HOLD}$ by writing to the PSPRDY bit in the PSPCFG register.

The Parallel Slave Peripheral does not generate interrupts by itself. Software is required to enable port pin RB7 (the $\overline{CS}$ input) as a falling-edge interrupt input for the Parallel Slave Peripheral to function. The $\overline{CS}$ signal must go high, then back low, for each data transfer. RB6 (the R/$\overline{W}$ input) must also be configured as an input. The setting in the RBDIR register for RB5 (the $\overline{HOLD}$ output) is overridden by the programming of the Parallel Slave Peripheral.

5.10.1 PSPCFG Register

The PSPCFG register is used to enable the Parallel Slave Peripheral, select which ports are used for data transfer, enable the $\overline{HOLD}$ output, and release the $\overline{HOLD}$ output when the data transfer is ready to complete.

7	6	5	4	3	2	1	0
PSPEN2	PSPEN1	PSPHEN	PSPRDY	Res		WD	BO

Table 5-19 PSPCFG Register

- *PSPEN2*—set to enable Port D for data transfer, clear to disable. (If this bit is set, the Parallel Slave Peripheral overrides the RDDIR register.)
- *PSPEN1*—set to enable Port C for data transfer, clear to disable. (If this bit is set, the Parallel Slave Peripheral will immediately override the RCDIR register.)
- *PSPHEN*—set to enable $\overline{HOLD}$ output, clear to disable. (If this bit is set, the Parallel Slave Peripheral will immediately override bit 5 of the RBDIR register.)
- *PSPRDY*—set to release $\overline{HOLD}$. This bit always reads as 0.
- *WD*—Watchdog time-out bit. Set at reset, if reset was triggered by Watchdog Timer overflow, otherwise cleared.
- *BO*—Brown-out reset bit. Set at reset, if reset was triggered by brown-out voltage level detection, otherwise cleared.

5.11 External Memory Interface

Port C and Port D can also be used for a parallel interface for up to 128K bytes of linear-addressed external memory, (not program memory) as shown in Figure 5-26. With additional software-based addressing on I/O, up to 2M bytes is possible. Port C implements the high address bits, and Port D is multiplexed between data and the low address bits. A level-triggered 8-bit latch (TI part number SN74LVC573A or equivalent) is required for demultiplexing. This latch passes the RD7:0 data when $\overline{LE}$ is high, and holds the data when $\overline{LE}$ is low.

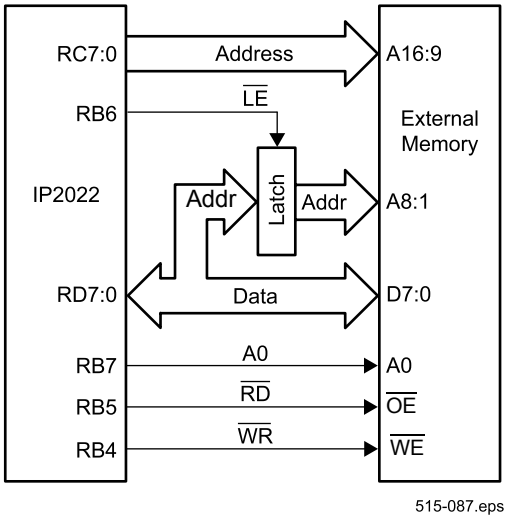


Figure 5-26 External Memory Interface

External memory is accessed as 16-bit words at word-aligned byte addresses 0x800000 to 0x81FFFE. External 8-bit memory can only be accessed through the current ADDRX/ADDRH/ADDRL pointer using the **`iread`** and **`iwrite`** instructions. Programs cannot execute directly out of external memory, and commands on the ISD/ISP interface cannot directly access external memory.

Note: In order to use the external memory interface correctly, RB[4:7], RC[0:7] and RD[0:7] must ALL be configured as outputs through their respective port-direction configuration registers.

5.11.1 EMCFG Register

Refer to Section 7.1.4 for field definitions and other information about the EMCFG register.

Note: When external memory is enabled (EMEN = 1), the RDDIR register value is overridden. PSP function will need to be disabled. Port B bits 4-7 interrupts need to be disabled.

Note: Wait one cycle after changing ADDRX bit 7, EMCFG bit 7, or ADDRSEL before executing an **`iread`** or **`iwrite`**. The ADDRX and ADDRSEL rules can be ignored if not switching between memory blocks.

Note: IREAD read cycle must complete before writing to DATAH or DATAL. This can be ignored if software is not writing to DATAH while hardware is writing to DATAL.

A read cycle to external memory has the timing shown in Figure 5-27. Write cycle timing is shown in Figure 5-28. The timings shown are recommended for 10ns and 12ns SRAMs. All external memory cycles are 16-bit transfers, with the low byte (A0 = 0) followed by the high byte (A0 = 1).

The number of system clocks required for one read or write access cycle is programmable to meet the SRAM timing. Figure 5-27 shows a typical SRAM read access cycle, with EMCFG register set to C9h, that is used to access 10ns SRAM with a 120 MHz system clock. SRAM grade reflects SRAM access time (Taa) in nanoseconds and typically is a last digit of an SRAM part number. SRAM has several important parameters which should be taken into account when calculating EMCFG register settings (refer to Table 5-21).

Table 5-20 shows maximum capacitance allowed on any SRAM controller signal line *vs*. a given SRAM grade and EMCFG setting that provides reliable access.

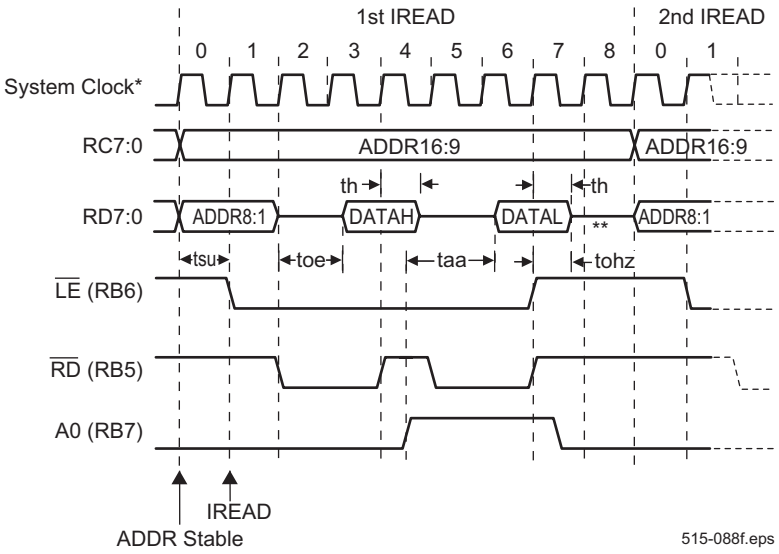
Table 5-20 EMCFG Settings

System CLK	Max Load Capacitance on $\overline{LE}$, RD, $\overline{WR}$, A0, and RD7:0(pF)	Tipd (ns)	Taa (ns)	EMCFG Register Setting
120	10	8	12	C9h
120	30	9	12	C9h
120	50	10	10	C9h

Note: The formulas and tables above assume that the address latch chip propagation delay is < Tsys_clk.

Software is responsible for allowing a memory cycle to complete before reading DATAH/DATAL registers by inserting instructions as follows:

- Instructions between IREAD and read of DATAH = EMRDT + 1
- Instructions between IREAD and read of DATAL = 2 * (EMRDT + 1)
- Instructions between consecutive IREADs (not including IREAD access cycles) =2 * (EMRDT + 1)



* System Clock is the IP2022's internal clock, and is shown for reference only.

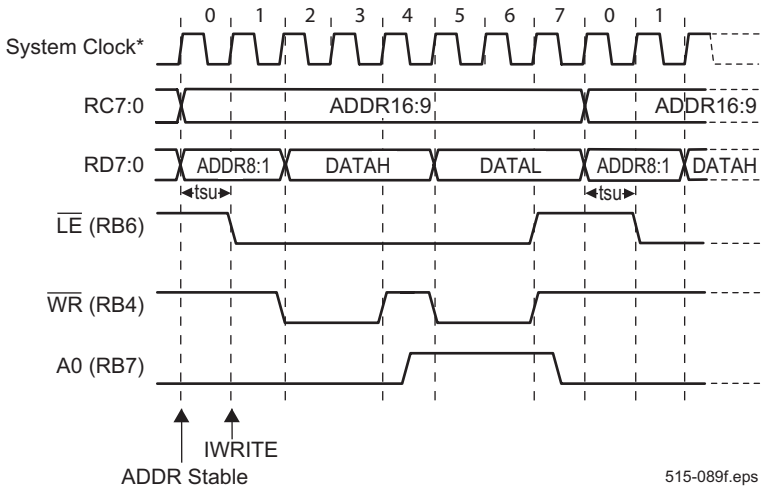
** Setting EMBRT adds one extra Z state cycle on the RD7:0 bus between cycle 7 and cycle 0, and increases the IREAD cycle time from 8 to 9 cycles.

Figure 5-27 SRAM Read Cycle (EMBRT=1; EMRDT=2 cycles)

Table 5-21 Timing Requirements for SRAM

IP2022's Timing Requirements	Signal Name	Units	Note
Data Hold Time	th ≥ 0	ns	
Set EMRDT and EMBRT in the EMCFG Register such that:			
SRAM Signal Description	**Signal Name**	**Units**	**Note**
Output Enable to Output Data	toe ≤ (EMRDT+1)*Tsys_clk - tipd	ns	1,2
Address to Output Data	taa ≤ (EMRDT+1.5)*Tsys_clk - tipd	ns	1,2
Output Disable to Output in Hi-Z	tohz ≤ (EMBRT+1)*Tsys_clk - 6ns	ns	1,2,3
Add instruction(s) after any ADDR change and before the IREAD or IWRITE in software, such that:			
Latch Signal Description	**Signal Name**		
Setup Time of Address Before $\overline{LE}$	tsu ≤ (# of instructions added + 1) * Tsys_clk - 3ns		

Notes:
1. IP2022's $\overline{RD}$ is connected to the external SRAM's $\overline{OE}$ pin.
2. Tsys_clk is the period of the IP2022 system clock, and tipd is the propagation delay internally in the IP2022, of the SRAM control signals (see Table 5-20).
3. The IP2022 should not output on RD7:0 before the SRAM tri-states this bus (set EMBRT = 1 if bus contention can exist with EMBRT = 0).



* System Clock is the IP2022's internal clock, and is shown for reference only.

Figure 5-28 SRAM Write Cycle (EMWRT=2 cycles)

6.0 In-System Programming

The IP2022 provides a dedicated serial interface for in-system programming (ISP) of the flash program memory and configuration block. ISP allows designers to incorporate a small connector which can be used to interface to a device programmer for programming or reprogramming the part after it has been soldered to a circuit board.

The interface used for in-system programming (ISP) and in-system debugging (ISD) is compatible with the SPI serial interface protocol. Whenever possible, a standard connector should be incorporated in the system design for in-system debugging and programming. The recommended connector layout for the ISD/ISP interface is shown in Figure 6-1. The connector is a male 10-pin connector with 100-mil pin spacing, whose pin assignments are listed in Table 6-1. The connector is keyed to prevent backward insertion.

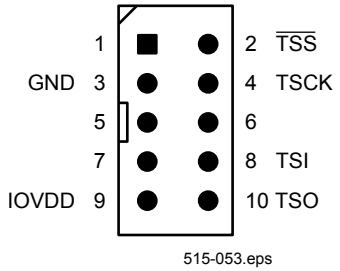


Figure 6-1 ISD/ISP Connector

Signal levels on the connector are LVTTL-compatible. The target system provides the TSCK, TSI, and $\overline{TSS}$ signals with 10K ohm pullup resistors.

Contact Ubicom for tools that program IP2022 devices.

Table 6-1 Connector Pin Assignments

Pin	Name	Description
1	KEY	Key (not a signal)
2	$\overline{TSS}$	*Target Slave Select*—Active-low signal which enables the IP2022 to communicate on the SPI bus. Connect to pin 1 on the IP2022.
3	GND	Ground
4	TSCK	*Target Data Clock*—Serial clock. Connect to pin 2 on the IP2022.
5	Reserved	
6	Reserved	

Table 6-1 Connector Pin Assignments (continued)

Pin	Name	Description
7	Reserved	
8	TSI	*Target Serial Input*—Sampled on the rising edge of TSCK. Connect to pin 3 on the IP2022.
9	IOVDD	Power. 2.3 - 3.6V (optional)
10	TSO	*Target Serial Output*—Driven by the IP2022 after the falling edge of TSCK. Connect to pin 4 on the part. The IP2022 drives this pin only if $\overline{TSS}$ is held low (TSO is tri-stated otherwise). The TSO pin is driven low if $\overline{TSS}$ is driven low while the part is in reset; TSO will be driven high as soon as the part is out of reset.

7.0 Memory Reference

7.0.1 Registers (sorted by address)

Table 7-1 shows the addresses and reset values of all special-purpose registers in data memory, sorted by their address.

Table 7-1 Register Addresses and Reset State

Address	Name	Description	Register Status Following Reset (Power-On, RST, Brown-Out RST, Watchdog RST)
0x001	Reserved	Reserved	Reserved
0x002	ADDRSEL	Selector for current external/program memory ADDRX/ADDRH/ADDRL	0000 0000
0x003	ADDRX	External/program memory pointer (bits 23:16)	0000 0000
0x004	IPH	Indirect Data RAM Pointer (high byte)	0000 0000
0x005	IPL	Indirect Data RAM Pointer (low byte, see Section 4.1)	0000 0000
0x006	SPH	Data RAM Stack Pointer (high byte)	0000 0000
0x007	SPL	Data RAM Stack Pointer (low byte, see Section 4.1)	0000 0000
0x008	PCH	Current PC (program counter) bits 15:8 (read-only)	1111 1111
0x009	PCL	Virtual register for direct PC modification	1111 0000
0x00A	WREG	W (working) register	0000 0000
0x00B	STATUS	STATUS register	1110 0000
0x00C	DPH	Data Pointer (high byte)	0000 0000
0x00D	DPL	Data Pointer (low byte, see Section 4.1)	0000 0000
0x00E	SPDREG	Current speed (read-only, see Section 3.5)	1001 0011
0x00F	MULH	Multiply result (high byte)	0000 0000
0x010	ADDRH	External/program memory address (bits 15:8)	0000 0000
0x011	ADDRL	External/program memory address (bits 7:0, see Section 5.11)	0000 0000
0x012	DATAH	External/program memory data (high byte)	0000 0000
0x013	DATAL	External/program memory data (low byte)	0000 0000
0x014	INTVECH	Interrupt vector (high byte)	0000 0000
0x015	INTVECL	Interrupt vector (low byte)	0000 0000
0x016	INTSPD	Interrupt speed register	0000 0000
0x017	INTF	Port B interrupt flags	Undefined
0x018	INTE	Port B interrupt enable bits	0000 0000
0x019	INTED	Port B interrupt edge select bits	0000 0000
0x01A	FCFG	Flash configuration register	0000 0000
0x01B	TCTRL	Timer 1/2 common control register	0000 0000

Table 7-1 Register Addresses and Reset State (continued)

Address	Name	Description	Register Status Following Reset (Power-On, RST, Brown-Out RST, Watchdog RST)
0x01C	XCFG	Extended configuration (bit 0 is read-only)	0000 000x (See Section 7.1.26 for FBUSY))
0x01D	EMCFG	External memory configuration register	0000 0000
0x01E	IPCH	Interrupt return address (high byte)	0000 0000
0x01F	IPCL	Interrupt return address (low byte)	0000 0000
0x020	RAIN	Data on Port A pins	N/A
0x021	RAOUT	Port A output latch	0000 0000
0x022	RADIR	Port A direction register	1111 1111
0x023	LFSRH	LFSR data register (high byte)	0000 0000
0x024	RBIN	Data on Port B pins	N/A
0x025	RBOUT	Port B output latch	0000 0000
0x026	RBDIR	Port B direction register	1111 1111
0x027	LFSRL	LFSR data register (low byte)	0000 0000
0x028	RCIN	Data on Port C pins	N/A
0x029	RCOUT	Port C output latch	0000 0000
0x02A	RCDIR	Port C direction register	1111 1111
0x02B	LFSRA	LFSR address register	0000 0000
0x02C	RDIN	Data on Port D pins	N/A
0x02D	RDOUT	Port D output latch	0000 0000
0x02E	RDDIR	Port D direction register	1111 1111
0x02F	Reserved	Reserved	Reserved
0x030	REIN	Data on Port E pins	N/A
0x031	REOUT	Port E output latch	0000 0000
0x032	REDIR	Port E direction register	1111 1111
0x033	Reserved	Reserved	Reserved
0x034	RFIN	Data on Port F pins	N/A
0x035	RFOUT	Port F output latch	0000 0000
0x036	RFDIR	Port F direction register	1111 1111
0x037	Reserved	Reserved	Reserved
0x038	Reserved	Reserved	Reserved
0x039	RGOUT	Port G output latch	0000 0000
0x03A	RGDIR	Port G direction register	1111 1111
0x03B	Reserved	Reserved	Reserved
0x03C	Reserved	Reserved	Reserved
0x03D	Reserved	Reserved	Reserved

Table 7-1 Register Addresses and Reset State (continued)

Address	Name	Description	Register Status Following Reset (Power-On, RST, Brown-Out RST, Watchdog RST)
0x03E	Reserved	Reserved	Reserved
0x03F	Reserved	Reserved	Reserved
0x040	RTTMR	Real-time timer value	0000 0000
0x041	RTCFG	Real-time timer configuration register	0000 0000
0x042	T0TMR	Timer 0 value	0000 0000
0x043	T0CFG	Timer 0 configuration register	0000 0000
0x044	T1CNTH	Timer 1 counter register high (read only)	0000 0000
0x045	T1CNTL	Timer 1 counter register low (read only)	0000 0000
0x046	T1CAP1H	Timer 1 Capture 1 register high (read only)	0000 0000
0x047	T1CAP1L	Timer 1 Capture 1 register low (read only)	0000 0000
0x048	T1CAP2H/T1CMP2H	Timer 1 Capture 2/Compare 2 register high	0000 0000
0x049	T1CAP2L/T1CMP2L	Timer 1 Capture 2/Compare 2 register low	0000 0000
0x04A	T1CMP1H	Timer 1 Compare 1 register high	0000 0000
0x04B	T1CMP1L	Timer 1 Compare 1 register low	0000 0000
0x04C	T1CFG1H	Timer 1 configuration register 1 high	0000 0000
0x04D	T1CFG1L	Timer 1 configuration register 1 low	0000 0000
0x04E	T1CFG2H	Timer 1 configuration register 2 high	0000 0000
0x04F	T1CFG2L	Timer 1 configuration register 2 low	0000 0000
0x050	ADCH	ADC value (high) (read only)	0000 0000
0x051	ADCL	ADC value (low) (read only)	0000 0000
0x052	ADCCFG	ADC configuration register	0000 0000
0x053	ADCTMR	ADC timer register	0000 0000
0x054	T2CNTH	Timer 2 counter register high (read only)	0000 0000
0x055	T2CNTL	Timer 2 counter register low (read only)	0000 0000
0x056	T2CAP1H	Timer 2 Capture 1 register high (read only)	0000 0000
0x057	T2CAP1L	Timer 2 Capture 1 register low (read only)	0000 0000
0x058	T2CAP2H/T2CMP2H	Timer 2 Capture 2/Compare 2 register high	0000 0000
0x059	T2CAP2L/T2CMP2L	Timer 2 Capture 2/Compare 2 register low	0000 0000
0x05A	T2CMP1H	Timer 2 Compare 1 register high	0000 0000
0x05B	T2CMP1L	Timer 2 Compare 1 register low	0000 0000
0x05C	T2CFG1H	Timer 2 configuration register 1 high	0000 0000
0x05D	T2CFG1L	Timer 2 configuration register 1 low	0000 0000
0x05E	T2CFG2H	Timer 2 configuration register 2 high	0000 0000
0x05F	T2CFG2L	Timer 2 configuration register 2 low	0000 0000
0x060	S1TMRH	SERDES 1 clock timer register (high bits)	0000 0000

Table 7-1 Register Addresses and Reset State (continued)

Address	Name	Description	Register Status Following Reset (Power-On, RST, Brown-Out RST, Watchdog RST)
0x061	S1TMRL	SERDES 1 clock timer register (low bits)	0000 0000
0x062	S1TBUFH	SERDES 1 transmit buffer (high bits)	Undefined
0x063	S1TBUFL	SERDES 1 transmit buffer (low bits)	Undefined
0x064	S1TCFG	SERDES 1 transmit configuration	0000 0000
0x065	S1RCNT	SERDES 1 received bit count (actual) (read-only)	0000 0000
0x066	S1RBUFH	SERDES 1 receive buffer (high bits) (read-only)	Undefined
0x067	S1RBUFL	SERDES 1 receive buffer (low bits) (read-only)	Undefined
0x068	S1RCFG	SERDES 1 receive configuration	0000 0000
0x069	S1RSYNC	SERDES 1 receive bit sync pattern	0000 0000
0x06A	S1INTF	SERDES 1 status/Interrupt flags	0000 0000
0x06B	S1INTE	SERDES 1 Interrupt enable bits	0000 0000
0x06C	S1MODE	SERDES 1 serial mode/clock select register	0000 0000
0x06D	S1SMASK	SERDES 1 receive sync mask	0000 0000
0x06E	PSPCFG	Parallel slave peripheral configuration register	0000 00xx (See Section 7.1.8 for BO, WD)
0x06F	CMPCFG	Comparator configuration register	0000 000X (See Section 7.1.3)
0x070	S2TMRH	SERDES 2 clock timer register (high bits)	0000 0000
0x071	S2TMRL	SERDES 2 clock timer register (low bits)	0000 0000
0x072	S2TBUFH	SERDES 2 transmit buffer (high bits)	Undefined
0x073	S2TBUFL	SERDES 2 transmit buffer (low bits)	Undefined
0x074	S2TCFG	SERDES 2 transmit configuration	0000 0000
0x075	S2RCNT	SERDES 2 received bit count (actual) (read-only)	0000 0000
0x076	S2RBUFH	SERDES 2 receive buffer (high bits) (read-only)	Undefined
0x077	S2RBUFL	SERDES 2 receive buffer (low bits) (read-only)	Undefined
0x078	S2RCFG	SERDES 2 receive configuration	0000 0000
0x079	S2RSYNC	SERDES 2 receive bit sync pattern	0000 0000
0x07A	S2INTF	SERDES 2 status/Interrupt flags	0000 0000
0x07B	S2INTE	SERDES 2 interrupt enable bits	0000 0000
0x07C	S2MODE	SERDES 2 serial mode/clock select register	0000 0000
0x07D	S2SMASK	SERDES 2 receive sync mask	0000 0000
0x07E	CALLH	Top of call stack (high 8 bits)	1111 1111

Table 7-1 Register Addresses and Reset State (continued)

Address	Name	Description	Register Status Following Reset (Power-On, RST, Brown-Out RST, Watchdog RST)
0x07F	CALLL	Top of call stack (low 8 bits)	1111 1111
0x080 to 0x0FF		Directly addressable general-purpose (global) registers	Undefined after power-on or brown-out reset, unchanged after RST or Watchdog Timer reset
0x100 to 0xFFF		Data memory RAM	Undefined after power-on or brown-out reset, unchanged after RST or Watchdog Timer reset

7.0.2 Program Memory

Table 7-2 shows the addresses and reset values of all program memory.

Table 7-2 Program Memory Addresses

Address	Description	Status Following Reset (Power-On, RST, Brown-Out RST, Watchdog RST)
0x0000 to 0x1FFF (word addresses)	Program Memory RAM (volatile memory)	Undefined after power-on or brown-out reset, unchanged after RST or Watchdog Timer reset
0x8000 to 0xFFFF (word addresses)	Program Memory Flash. Factory programmed to all FFFF (fully erased). This is non-volatile memory (won't change after power down).	Unchanged after power-on, brown-out reset, RST or Watchdog Timer reset (changes only during ISP programming, and during flash self-programming)
0x10000 to 0x1003F (word addresses)	Flash Configuration Block (see Section 3.10). Non-volatile. Flash Configuration is factory programmed to: **Word Address** $ 10000 FUSE0 = 1000 $ 10001 FUSE1 = FFF7 $ 10004 TRIM0 = FBFE	Unchanged after power-on, brown-out reset, RST or Watchdog Timer reset (changes only during ISP programming, and during flash self-programming)

7.1 Register Bit Definitions

For those registers which have special functions assigned to bits or fields within the register, the definition of those bits and fields is described below. The registers are presented alphabetically.

7.1.1 ADCCFG Register

A/D converter configuration.

7	6	5	4	3	2	0
ADCREF	ADCJST		Rsrvd.	ADCGO	ADCS2:0	

Name	Description
ADCREF	A/D converter reference voltage select 0 = AVdd is the reference voltage 1 = RG3 port pin is used to receive an external reference voltage
ADCJST	A/D converter result justification mode select 00 = Right justified 01 = Signed 10 = Left justified 11 = Reserved
ADCGO	A/D converter GO/DONE bit 0 = When the last conversion has completed, this bit reads as 0. 1 = Write 1 to begin a new conversion. While the conversion is in progress, this bit reads as 1.
ADCS2:0	A/D converter input channel select 000 = Port pin RG0 001 = Port pin RG1 010 = Port pin RG2 011 = Port pin RG3 100 = Port pin RG4 101 = Port pin RG5 110 = Port pin RG6 111 = Port pin RG7

7.1.2 ADCTMR Register

The ADCTMR register is used to specify the number of system clock cycles required for a delay of 1736 ns, which is used to provide the 1.152MHz (48 kHz x 24) clock period reference clock for the A/D converter.

7.1.3 CMPCFG Register

Comparator configuration.

7	6	5	4	3	2	1	0
CMPEN	CMPOE	CMPHYS	Reserved				CMPRES

Name	Description
CMPEN	Comparator enable bit 0 = Comparator disabled 1 = Comparator enabled
CMPOE	Comparator output enable bit 0 = Comparator output disabled. 1 = Comparator output enabled on port pin RG0.
CMPHYS	Comparator hysteresis enable bit 0 = Hysteresis disabled 1 = Hysteresis enabled
CMPRES	Comparator result (read-only) 0 = RG2 voltage < RG1 1 = RG1 voltage < RG2
Note that the comparator bandwidth (shown in “Comparator DC and AC Specifications) is reduced when hysteresis is enabled.	

7.1.4 EMCFG Register

External memory interface configuration.

7	6	5 3	2 0
EMEN	EMBRT	EMWRT2:0	EMRDT2:0

Name	Description
EMEN	Enable external memory interface 0 = Port C, Port D and RB7:4 available for general-purpose I/O 1 = Port C, Port D and RB7:4 used for external memory interface
EMBRT	Enable bus release wait state 0 = No wait state 1 = One wait state cycle added to IREAD and IWRITE after read of DATAL and before ADDR8:1 is put on RD7:0 bus
EMWRT2:0	$\overline{WR}$ pulse width, in system clock cycles 000 = 1 100 = 5 001 = 2 101 = 6 010 = 3 110 = 7 011 = 4 111 = 8
EMRDT2:0	$\overline{RD}$ pulse width, in system clock cycles 000 = 1 100 = 5 001 = 2 101 = 6 010 = 3 110 = 7 011 = 4 111 = 8

7.1.5 FCFG Register

Flash configuration.

7	6	5	4	3	2	1	0
FRDTS1:0		FRDTC1:0		FWRT3:0			

Name	Description		
FRDTS1:0	The core clock frequency is automatically reduced (if necessary) when executing out of flash memory to prevent the flash memory access time from being too short. The FRDTS1:0 bits specify the minimum number of system clock cycles required for instruction execution from flash memory. The actual execution speed from flash memory will be the slower of the speed indicated in the SPDREG register and the speed specified by the FRDTS1:0 bits. The 11 setting can always be used, but it may cause slower flash operation than necessary.		
	Set to this:	**If System Clock Frequency (MHz) is**	**System Clock Cycles For Each Flash Instruction Cycle**
	00	0–40	1 cycle
	01	40–80	2 cycles
	10	80–120	3 cycles
	11	reserved	4 cycles
Note	Flash instruction execution = 25ns minimum		
FRDTC1:0	To prevent the flash memory access time from being too short, it is necessary to specify the number of CPU core cycles between reading the flash memory using an **`fread`** instruction (or an **`iread`** or **`ireadi`** instruction while executing from RAM to read flash) and the time that DATAH and DATAL are written by the IP2022. Because the CPU core is subject to changes in speed, the value programmed in these bits should be appropriate for the fastest speed that might be used (typically, the faster of the main line code and the interrupt service routine). The FRDTC1:0 bits specify the number of CPU core clock cycles required for flash read access. Even with this configured, enough cycles must be executed after the flash read instruction and before reading DATAH or DATAL to account for the minimum flash access time.		
	Set to this:	**If CPU Core Frequency (MHz) is**	**Core Clock Cycles For Each Flash Read Cycle**
	00	**120 MHz Part**	1 cycle
	01	0–40	2 cycles
	10	40–80	3 cycles
	11	80–120 reserved	4 cycles
Note	**`fread`**/**`iread`**/**`ireadi`** of flash= 25ns minimum		

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>FWRT3:0</td><td>The flash memory ferase, fwrite and ISP flash write, flash block erase, or flash bulk erase timing is derived from the CPU core clock through a programmable divider. The FWRT3:0 bits specify the divisor. The time base must be 1 to 2 microseconds. Below 1 microsecond, the flash memory will be underprogrammed, and data retention is not guaranteed. Above 2 microseconds, the flash memory will be overprogrammed, and reliability is not guaranteed.
<table>
<tr><th>Set to this:</th><th>If CPU Core Frequency is:</th><th>FWRT Frequency Divisor</th></tr>
<tr><td>0000</td><td>1–2 MHz</td><td>2</td></tr>
<tr><td>0001</td><td>2–3 MHz</td><td>3</td></tr>
<tr><td>0010</td><td>3–4 MHz</td><td>4</td></tr>
<tr><td>0011</td><td>4–6 MHz</td><td>6</td></tr>
<tr><td>0100</td><td>6–8 MHz</td><td>8</td></tr>
<tr><td>0101</td><td>8–12 MHz</td><td>12</td></tr>
<tr><td>0110</td><td>12–16 MHz</td><td>16</td></tr>
<tr><td>0111</td><td>16–24 MHz</td><td>24</td></tr>
<tr><td>1000</td><td>24–32 MHz</td><td>32</td></tr>
<tr><td>1001</td><td>32–48 MHz</td><td>48</td></tr>
<tr><td>1010</td><td>48–64 MHz</td><td>64</td></tr>
<tr><td>1011</td><td>64–96 MHz</td><td>96</td></tr>
<tr><td>1100</td><td>96–128 MHz</td><td>128</td></tr>
<tr><td>1101</td><td>Reserved</td><td>192</td></tr>
<tr><td>1110</td><td>Reserved</td><td>256</td></tr>
<tr><td>1111</td><td>Reserved</td><td>384</td></tr>
</table></td></tr>
<tr><td>Note:</td><td>If FCFG & OSC1 are optimal:
fwrite = 42us
ferase = 20ms, because FPERT in TRIM0 should be 0</td></tr>
</table>

7.1.6 INTSPD Register

Configuration of clock and PLL settings to be used during an interrupt service routine. INTSPD is copied to SPDREG when an interrupt occurs. See Table 3-5 for **`reti`** options.

7	6	5	4	3	0
$\overline{\text{PLL}}$	$\overline{\text{OSC}}$	CLK1:0		CDIV3:0	

Name	Description
$\overline{\text{PLL}}$	Run-time control of PLL clock multiplier operation. If the PLL is not required, power consumption can be reduced by disabling it. 0 = PLL clock multiplier enabled 1 = PLL clock multiplier disabled
$\overline{\text{OSC}}$	Run-time control of OSC oscillator operation. If the crystal oscillator is not required, power consumption can be reduced by disabling it (stops OSC oscillator and blocks propagation of OSC1 external clock input). 0 = OSC oscillator enabled 1 = OSC oscillator disabled
CLK1:0	Selects the system clock source. 00 = PLL clock multiplier. Do not use if an interrupt can awaken the part from sleep (can use a **`speed`** instruction in the ISR instead). 01 = OSC oscillator/external clock on OSC1 input 10 = RTCLK oscillator/external clock on RTCLK1 input 11 = System clock disabled (off) Note: If the OSC crystal driver is stopped (SPDREG bit 6 = 1) and Port B or Real Time Timer interrupts are enabled, then INTSPD bits 5 and 4 must not both be 0, because the crystal startup time plus PLL startup time may be greater than WUDP2:0 (see Figure 3-15).
CDIV3:0	Selects the system clock divisor. 0000 = 1 1000 = 12 0001 = 2 1001 = 16 0010 = 3 1010 = 24 0011 = 4 1011 = 32 0100 = 5 1100 = 48 0101 = 6 1101 = 64 0110 = 8 1110 = 128 0111 = 10 1111 = System clock disabled (off)

7.1.7 LFSRA Register

Linear Feedback Shift Register configuration.

7	4	3	0
UNIT3:0		INDEX3:0	

Name	Description
UNIT3:0	LFSR unit number (only 0, 1, 2, and 3 are valid)
INDEX3:0	Index to the LFSR register being accessed (see Table 5-17)

7.1.8 PSPCFG Register

Parallel Slave Peripheral configuration and reset bits.

7	6	5	4	3	1	0
PSPEN2	PSPEN1	PSPHEN	PSPRDY	Res	WD	BO

Name	Description
PSPEN2	Port D enable bit 0 = Port D is available for general-purpose I/O 1 = Port D is configured for the Parallel Slave Peripheral interface
PSPEN1	Port C enable bit 0 = Port C is available for general-purpose I/O 1 = Port C is configured for the Parallel Slave Peripheral interface
PSPHEN	$\overline{\text{HOLD}}$ output enable bit 0 = $\overline{\text{HOLD}}$ output disabled. Port pin RB5 available for general-purpose I/O. 1 = $\overline{\text{HOLD}}$ output enabled on port pin RB5.
PSPRDY	Ready bit 0 = This bit always reads as zero. 1 = Write 1 to release $\overline{\text{HOLD}}$ when the IP2012 / IP2022 is ready to allow the data transfer to complete.
WD	Watchdog timeout bit. This bit is set by the processor when it goes through a watchdog reset. This bit is not cleared by a Watchdog reset, but is cleared by all other reset sources. Therefore, if using the Watchdog feature, put the following in your bootup routine: 1. If this WD bit is set, execute your desired Watchdog reset indication routine. 2. Set this WD bit. These two steps must be executed before the first **`cwdt`** instruction. Note: If WUDX in FUSE0 is more than 70ms longer than the watchdog timeout period in FUSE1, a Power-On Reset or a Brown-Out Reset may set this bit. (Do not use this bit if WUDX is more than 70ms longer than the watchdog timeout period).
BO	Brown-out reset bit. Set at reset, if reset was triggered by brown-out voltage level detection, otherwise cleared

7.1.9 RTCFG Register

Real-Time Timer configuration.

7	6 3	2	1	0
RTEN	RTPS3:0	RTSS	RTIE	RTIF

Name	Description
RTEN	Real-Time Timer enable bit 0 = Real-Time Timer disabled 0 = Real-Time Timer enabled
RTPS3:0	Real-Time Timer prescaler divisor 0000 = 1 1000 = 256 0001 = 2 1001 = 512 0010 = 4 1010 = 1024 0011 = 8 1011 = 2048 0100 = 16 1100 = 4096 0101 = 32 1101 = 8192 0110 = 64 1110 = 16384 0111 = 128 1111 = 32768
RTSS	Real-Time Timer clock source select 0 = external OSC clock 1 = external RTCLK clock
RTIE	Real-Time Timer interrupt enable bit 0 = Real-Time Timer interrupt disabled 1 = Real-Time Timer interrupt enabled
RTIF	Real-Time Timer interrupt flag 0 = No timer overflow has occurred since this bit was last cleared 1 = Timer overflow has occurred. This bit goes high two cycles after the actual overflow occurs.

7.1.10 SxINTE/SxINTF Register

Indicates the SERDES conditions that may be enabled as interrupts.

The SxINTE (S1INTE or S2INTE) register has the same format as the SxINTF register. For each condition indicated by a flag in the SxINTF register, setting the corresponding bit in the SxINTE register enables the interrupt for that condition.

7	6	5	4	3	2	1	0
RXERROR	RXEOP	SYND	TXBE	TXEOP	SXLINKPULSE	RXBF	RXXCRS

Name	Description
RXERROR	Receive error interrupt flag 10Base-T mode: Manchester encoding data phase error USB mode: bit unstuffing error (1111111 received) 0 = Receive error has not been detected since this bit was last cleared 1 = Receive error has been detected
RXEOP	End-of-Packet detection interrupt flag 10Base-T and USB modes: end-of-packet detected GPSI mode: RxEN deasserted SPI mode: Set on rising edge. 0 = End-of-Packet has not been detected since this bit was last cleared 1 = End-of-Packet has been detected
SYND	Synchronization pattern detection interrupt flag (10Base-T and USB modes only) 0 = Synchronization pattern has not been detected since this bit was last cleared 1 = Synchronization pattern has been detected
TXBE	Transmit buffer (SxTXBUF) empty interrupt flag 0 = Transmit buffer has not been empty since this bit was last cleared 1 = Transmit buffer has been empty
TXEOP	Transmit underrun. This bit is set when all the data has been transmitted and no new data has been loaded in the register. In USB and 10Base-T modes, this causes an EOP condition to be generated. 0 = Transmit underrun has not occurred since this bit was last cleared 1 = Transmit underrun has occurred
SXLINKPULSE	Set after a link pulse of 60 to 200 ns duration is detected in Ethernet mode. Also known as TxIdle in USB mode. 0 = 10Base-T mode: No link pulse has been detected since this bit was last cleared USB mode: SERDES is transmitting 1 = 10Base-T mode: Link pulse detected USB mode: SERDES is not transmitting
RXBF	Receive buffer full interrupt flag 0 = Receive buffer has not been full since this bit was last cleared 1 = Receive buffer has been full
RXXCRS	Set while the carrier is sensed. 10bT mode: clear to 0 0 = USB mode: RxBUSY is detected - SERDES is receiving 1 = USB mode: RxBUSY is not detected - SERDES is not receiving

7.1.11 SxMODE Register

SERDES protocol mode configuration.

7	6	5	4	3	2	1	0
REV	PRS2:0			SUBM1:0		CLKS1:0	

Name	Description
REV	Read-only as “1” in latest revision. Refer to IP2022 silicon errata sheet.
PRS2:0	Protocol select (see Table 5-3). All other encodings are reserved. 000 = Disabled 001 = 10Base-T 010 = USB Bus 011 = UART 101 = SPI 110 = GPSI
SUBM1:0	Submode select USB mode: 01 = Low-speed USB interface 10 = High-speed USB interface SPI mode: 00 = Positive clock polarity, receive on rising edge, transmit on falling edge 01 = Positive clock polarity, receive on falling edge, transmit on rising edge 10 = Negative clock polarity, receive on falling edge, transmit on rising edge 11 = Negative clock polarity, receive on rising edge, transmit on falling edge GPSI mode: 00 = Receive on rising edge, transmit on falling edge 01 = Receive on falling edge, transmit on falling edge 10 = Receive on rising edge, transmit on rising edge 11 = Receive on falling edge, transmit on rising edge
CLKS1:0	SerDes Clock source select (see Figure 5-8 and Figure 3-16). 00 = Clock disabled 01 = SxCLK input 10 = OSC clock oscillator 11 = post-PLL clock Note: When switching CLKS1:0 to 10, a delay is needed before reliable writes to SERDES registers can be made. The required delay, in number of core instructions, is (core clock frequency / new SERDES clock frequency) x 2.

7.1.12 SxRCFG Register

SERDES RX shift count, USB sync detect and data polarity configuration.

7	6	5	4	0
MASSEL	SYNCDETEN	RPOREV	RXSCNT4:0	

Name	Description
MASSEL	10Base-T mode: 0 = Normal polarity detected 1 = Reverse polarity detected GPSI or SPI mode: 0 = Slave mode 1 = Master mode
SYNCDETEN	Synchronization byte detection enable (USB mode only) 0 = Synchronization byte detection enabled 1 = Synchronization byte detection disabled
RPOREV	Receive data polarity reversal select 0 = Data polarity uninverted 1 = Data polarity inverted
RXSCNT4:0	Receive shift count, specifies number of bits to receive

7.1.13 SxRCNT Register

SERDES RX activity configuration.

7	6	5	4 0
BITORDER	RxCRSED	RxCRS	RXACNT4:0

Name	Description
BITORDER	Bit order for transmit and receive 0 = LSB first 1 = MSB first
RxCRSED	Not used for 10baseT mode. For GPSI Slave mode: 0 = Disable TxBUSY input 1 = Enable TxBUSY input
RxCRS	Carrier Sense Status: Current state of carrier
RXACNT4:0	Receive shift count, actual number of bits received (read-only). Exceptions occur during the last transfer: RXACNT = 0 if bit count is less than 8 RXACNT = 8 if bit count is greater than or equal to 8, but less than 16 RXACNT = 16 if bit count is greater than or equal to 16 and the RXSCNT4:0 field in the SxRCFG register is 16

7.1.14 SxRSYNC Register

SERDES sync pattern configuration.

7 2	1	0
SYNCPAT7:2	SQUELCHEN	DRIBBITEN

Name	Description
SYNCPAT7:2	Synchronization pattern, bits 7:2 (USB mode only)
SQUELCHEN	USB mode: synchronization pattern, bit 1 10Base-T mode: 0 = Squelch disabled 1 = Squelch enabled All other modes: 0
DRIBBITEN	USB mode: synchronization pattern, bit 0 10Base-T mode: 0 = Hardware handles dribble bit 1 = Software is responsible for handling dribble bit

7.1.15 SxSMASK Register

SERDES sync-pattern configuration.

10Base-T mode:

7	6	3	2	1	0
Resrvd.	PREAMCNT3:0		Resrvd.	CONTPAIR	Resrvd.

USB mode:

7	0
MASK7:0	

Name	Description
PREAMCNT3:0	Preamble pair count (10Base-T mode only). All other encodings are reserved. 0000 = 24 pairs 0001 = 20 pairs 0010 = 16 pairs 0011 = 12 pairs 0100 = 8 pairs 0101 = 4 pairs
CONTPAIR	Configures the detection of consecutive pairs of “10” for sync detection. 0 = Sync detected if 6 “10” pairs + “11”. 1 = Use PREAMCNT for number of “10” pairs
MASK7:0	Mask bits for SxRSYNC (USB mode only) 0 = Ignore corresponding bit in SxRSYNC 1 = Use corresponding bit in search pattern for synchronization byte

7.1.16 SxTCFG Register

SERDES TX shift count configuration.

7	6	5	4	0
GLOBEN	LPBACK	TPOREV	TXSCNT4:0	

Name	Description
GLOBEN	Global enable bit 0 = Disable SERDES output 1 = Enable SERDES output (must use for TX). If enabling SERDES1, the REOUT port data are overridden by SERDES1 outputs. If enabling SERDES2, the RFOUT port data are overridden by SERDES2 outputs.
LPBACK	Loopback enable bit 0 = Normal operation 1 = Output is driven into input
TPOREV	Transmit data polarity reversal select (UART mode only) 0 = Data polarity uninverted 1 = Data polarity inverted
TXSCNT4:0	Transmit shift count, specifies number of bits to transmit

7.1.17 SxTMRH/SxTMRL Register

Used to specify the divide value for the OSC clock, post-PLL clock or SxCLK input (specified in the SxMode register bits CLKS1:0, Section 7.1.11) to generate the SERDES clock. The effective divide value = {SxTMRH, SxTMRL} + 1, except, in the case of SPI and GPSI Master, the effective divide value = {(SxTMRH/SxTMRL) + 1} x 2.

7.1.18 SPDREG Register

Status of clock and PLL settings during run-time.

Note: This is a read-only register, use **speed** instruction to change settings.

7	6	5	4	3	0
$\overline{\text{PLL}}$	$\overline{\text{OSC}}$	CLK1:0		CDIV3:0	

Name	Description
$\overline{\text{PLL}}$	Run-time control of PLL clock multiplier operation. If the PLL is not required, power consumption can be reduced by disabling it, but a WUDP delay is required to start it again (controlled in FUSE0). 0 = PLL clock multiplier on 1 = PLL clock multiplier off
$\overline{\text{OSC}}$	Run-time control of OSC oscillator operation. If the OSC clock is not required, power consumption can be reduced by disabling it (stops OSC crystal oscillator and blocks propagation of OSC1 external clock input). 0 = OSC oscillator enabled 1 = OSC oscillator disabled
CLK1:0	Selects the system clock source. 00 = PLL clock multiplier 01 = OSC oscillator/external clock on OSC1 input 10 = RTCLK oscillator/external clock on RTCLK1 input 11 = System clock disabled (off)
CDIV3:0	Selects the divisor which divides the system clock to give the core clock. 0000 = 1 0001 = 2 0010 = 3 0011 = 4 0100 = 5 0101 = 6 0110 = 8 0111 = 10 1000 = 12 1001 = 16 1010 = 24 1011 = 32 1100 = 48 1101 = 64 1110 = 128 1111 = Core clock disabled (off)

7.1.19 STATUS Register

Condition flags for the results of arithmetic and logical operations, the page bits, and bits which indicate the skipping state of the core and control of continuation skip after return from interrupt.

7 5	4	3	2	1	0
PA2:0	SAR	SSF	Z	DC	C

Name	Description
PA2:0	Program memory page select bits (read only). Used to extend the 13-bit address encoded in jump and call instructions (these 3 bits are written to the upper 3 bits of the program counter when a jump or call occurs). Modified using the **`page`** instruction.
SAR	Skip After Return bit. Indicates if the core should be in the skipping/not state after the completion of a return instruction (**`ret`**, **`retnp`**, or **`retw`** instructions, but not **`reti`**). The return instruction will also clear the SAR bit to ensure correct behavior after the dynamic jump. 0 = The core should not be in a skipping state upon completion of the return. 1 = The core should be in a skipping state upon completion of the return.
SSF	Shadowed Skipping/not state Flag. Gives the ISR the ability to know if the interrupt occurred immediately following a skip instruction. The software can choose either to clear the SSF flag in the ISR or to make the first instruction of the mainline context switching code a **`nop`** to flush out the skip state. 0 = The core was not in a skipping state when interrupted. 1 = The core was in a skipping state when interrupted.
Z	Zero bit. Affected by most logical, arithmetic, and data movement instructions (refer to “Flags Affected” column in Table 4-2 through Table 4-7). Set if the result was zero, otherwise cleared. 0 = Result of last ALU operation was non-zero. 1 = Result of last ALU operation was zero.
DC	Digit Carry bit. After addition, set if carry from bit 3 occurred, otherwise cleared. After subtraction, cleared if borrow from bit 3 occurred, otherwise set. 0 = Last addition did not generate carry out of bit 3, or last subtraction generated borrow out of bit 3. 1 = Last addition generated carry out of bit 3, or last subtraction did not generate borrow out of bit 3.
C	Carry bit. After addition, set if carry from bit 7 of the result occurred, otherwise cleared. After subtraction, cleared if borrow from bit 7 of the result occurred, otherwise set. After rotate (**`rr`** or **`rl`**) instructions, loaded with the LSB or MSB of the operand, respectively. 0 = Last addition did not generate carry out of bit 7, last subtraction generated borrow out of bit 7, or last rotate loaded a 0. 1 = Last addition generated carry out of bit 7, last subtraction did not generate borrow out of bit 7, or last rotate loaded a 1.

When writing directly to the STATUS register, it is recommended to use only setb and clrb commands, since writing to the whole STATUS register at once may produce unexpected results. For example, a CLR STATUS command may set the Z bit, and leave the DC and C bits unchanged.

7.1.20 T0CFG Register

Timer 0 configuration.

7	6 3	2	1	0
T0EN	T0PS3:0	Rsrvd.	T0IE	T0IF

Name	Description
T0EN	Enables Timer 0 0 = Timer 0 disabled 1 = Timer 0 enabled
T0PS3:0	Specifies Timer 0 prescaler divisor 0000 = 1 0001 = 2 0010 = 4 0011 = 8 0100 = 16 0101 = 32 0110 = 64 0111 = 128 1000 = 256 1001 to 1111 = Reserved
T0IE	Timer 0 interrupt enable bit 0 = Timer 0 interrupt disabled 1 = Timer 0 interrupt enabled
T0IF	Timer 0 interrupt flag 0 = No timer overflow has occurred since this bit was last cleared 1 = Timer overflow has occurred

7.1.21 TxCFG1H Register

Timer 1 and 2 configuration.

7	6	5	4	3	2	1	0
OFIE	CAP2IE CMP2IE	CAP1IE	CMP1IE	OFIF	CAP2IF CMP2IF	CAP1IF	CMP1IF

Name	Description
OFIE	Timer overflow interrupt enable bit 0 = Overflow interrupt disabled 1 = Overflow interrupt enabled
CAP2IE or CMP2IE	PWM mode: Compare 2 interrupt enable bit Capture/Compare mode: Capture 2 interrupt enable bit 0 = Capture/Compare 2 interrupt disabled 1 = Capture/Compare 2 interrupt enabled
CAP1IE	Capture 1 interrupt enable bit 0 = Capture 1 interrupt disabled 1 = Capture 1 interrupt enabled
CMP1IE	Compare 1 interrupt enable bit 0 = Compare 1 interrupt disabled 1 = Compare 1 interrupt enabled
OFIF	Timer overflow interrupt flag 0 = No timer overflow has occurred since this bit was last cleared 1 = Timer overflow has occurred
CAP2IF or CMP2IF	PWM mode: Compare 2 interrupt flag (i.e. timer value matched TxCMP2 value) Capture/Compare mode: Capture 2 flag (i.e. TxCPI2 input triggered) 0 = No capture/compare 2 event has occurred since this bit was last cleared 1 = Capture/compare 2 event has occurred
CAP1IF	Capture 1 interrupt flag 0 = No capture 1 event has occurred since this bit was last cleared 1 = Capture 1 event has occurred
CMP1IF	Compare 1 interrupt flag 0 = No compare 1 event has occurred since this bit was last cleared 1 = Compare 1 event has occurred

7.1.22 TxCFG2H Register

Timer 1 and 2 configuration.

7	6	5	4	3	0
0	0	0	0	PS3:0	

Name	Description		
PS3:0	Timer prescaler divisor		
	0000 = 1	1000 = 256	
	0001 = 2	1001 = 512	
	0010 = 4	1010 = 1024	
	0011 = 8	1011 = 2048	
	0100 = 16	1100 = 4096	
	0101 = 32	1101 = 8192	
	0110 = 64	1110 = 16384	
	0111 = 128	1111 = 32768	

7.1.23 TxCFG1L Register

Timer 1 and 2 configuration.

7	6	5	4	3	2	1	0
MODE	OEN	ECLKEN	CPI2EN	CPI1EN	ECLKEDG	CAP1RST	TMREN

Name	Description
MODE	Timer mode select 0 = PWM/timer mode 1 = Capture/compare mode
OEN	TxOUT enable bit 0 = TxOUT disabled. Port pin available for general-purpose I/O. 1 = TxOUT enabled. Port pin must be configured for output in corresponding RxDIR register bit. Output is on RA3 for T1, RB3 for T2
ECLKEN	TxCLK enable bit 0 = TxCLK disabled. Port pin available for general-purpose I/O. 1 = TxCLK enabled as clock source for timer. Enabling this bit does not make any other restrictions on the use of the TxCLK port pin for general-purpose I/O.
CPI2EN	TxCPI2 enable bit 0 = System clock enabled as clock source for timer. TxCPI2 port pin available for general-purpose I/O. 1 = TxCLK enabled as clock source for timer. Enabling this bit does not make any other restrictions on the use of the port pin for general-purpose I/O.
CPI1EN	TxCPI1 enable bit 0 = Capture 1 input disabled. TxCPI1 port pin available for general-purpose I/O. 1 = TxCPI1 enabled as capture 1 input. Enabling this bit does not make any other restrictions on the use of the port pin for general-purpose I/O.
ECLKEDG	TxCLK edge sensitivity select. (This bit is ignored if the ECLKEN bit is clear.) 0 = TxCLK increments timer on rising edge 1 = TxCLK increments timer on falling edge
CAP1RST	Reset timer on capture 1 event enable bit 0 = Timer value unchanged by occurrence of a capture 1 event 1 = Timer value cleared by occurrence of a capture 1 event
TMREN	Timer enable bit 0 = Timer disabled. Timer clock source shut off to reduce power consumption. 1 = Timer enabled

7.1.24 TxCFG2L Register

Timer 1 and 2 configuration.

7	6	5	4	3	2	1	0
Reserved	TOUTSET	TOUTCLR	CPI2CPI1	CPI2EDG1:0		CPI1EDG1:0	

Name	Description
TOUTSET	Override bit to set the TxOUT output. This bit always reads as zero. 0 = Writing 0 to this bit has no effect 1 = Writing 1 to this bit forces the TxOUT signal high
TOUTCLR	Override bit to clear the TxOUT output. This bit always reads as zero. 0 = Writing 0 to this bit has no effect 1 = Writing 1 to this bit forces the TxOUT signal low
CPI2CPI1	Internally connect the TxCPI2 input to the TxCPI1 input. This makes the TxCPI2 port pin available for general-purpose I/O. 0 = No internal connection between TxCPI1 and TxCPI2 1 = TxCPI1 and TxCPI2 internally connected
CPI2EDG1:0	TxCPI2 edge sensitivity select 00 = Falling edge on TXCPI2 recognized as capture 2 event 01 = Rising edge on TXCPI2 recognized as capture 2 event 10 = Any falling or rising edge on TXCPI2 recognized as capture 2 event 11 = Any falling or rising edge on TXCPI2 recognized as capture 2 event
CPI1EDG1:0	TxCPI1 edge sensitivity select 00 = Falling edge on TXCPI1 recognized as capture 1 event 01 = Rising edge on TXCPI1 recognized as capture 1 event 10 = Any falling or rising edge on TXCPI1 recognized as capture 1 event 11 = Any falling or rising edge on TXCPI1 recognized as capture 1 event

7.1.25 TCTRL Register

Timer 1 and 2 configuration.

7	6	5	4	3	2	1	0
0	0	T2IE	T1IE	0	0	T2RST	T1RST

Name	Description
T2IE	Timer 2 interrupt enable 0 = Timer 2 interrupt disabled 1 = Timer 2 interrupt enabled
T1IE	Timer 1 interrupt enable 0 = Timer 1 interrupt disabled 1 = Timer 1 interrupt enabled
T2RST	Timer 2 reset bit. This bit always reads as zero. 0 = Writing 0 to this bit has no effect. 1 = Writing 1 to this bit clears Timer 2.
T1RST	Timer 1 reset bit. This bit always reads as zero. 0 = Writing 0 to this bit has no effect. 1 = Writing 1 to this bit clears Timer 1.

7.1.26 XCFG Register

Extra configuration bits for various functions.

7	6	5	4	3	2	1	0
GIE	$\overline{\text{FWP}}$	RTEOS	$\overline{\text{RTOSC_EN}}$	INT_EN	Rsvd		FBUSY

Name	Description
GIE	Global interrupt enable bit 0 = Interrupts disabled 1 = Interrupts enabled
$\overline{\text{FWP}}$	Flash write protect bit. This bit only affects operation of **`fwrite`** and **`ferase`** self-programming instructions on flash, not programming through the ISD/ISP interface. Does not affect writes or erases of RAM. 0 = Writes to flash memory disabled (act as **`nop`** instructions) 1 = Writes to flash memory enabled
RTEOS	Real-time timer oversampling bit 0 = Oversampling disabled 1 = Oversampling enabled
$\overline{\text{RTOSC_EN}}$	RTCLK oscillator enable bit 0 = RTCLK oscillator is operational 1 = RTCLK oscillator turned off
INT_EN	**`int`** instruction interrupt enable bit 0 = **`int`** instructions only increment the PC, like **`nop`** 1 = **`int`** instructions cause interrupts
FBUSY	Flash memory busy bit (read-only). For more information about programming the flash memory, see Section 4.7. 0 = Flash memory is idle 1 = Fetching instructions out of flash memory or busy processing an **`iread`**, **`ireadi`**, **`fwrite`**, **`fread`** or **`ferase`** instruction on Flash

8.0 Electrical Characteristics

8.1 Absolute Maximum Ratings

These are ratings beyond which permanent damage may occur. Correct operation is not guaranteed outside of the DC specifications in Section 8.2.

Parameter	Minimum	Maximum	Units
Ambient temperature under bias	-40	85	°C
Storage temperature	-65	150	°C
Voltage on DVdd, XVdd, AVdd, and GVdd with respect to Vss	-0.5	3.5	V
Voltage on IOVdd with respect to Vss	-0.5	4.5	V
Voltage on Port A through Port F, OSC1, $\overline{RST}$, RTCLK1, TSCK, TSI, and $\overline{TSS}$ inputs with respect to Vss	-0.5	5.7	V
Voltage on Port G inputs with respect to Vss	-0.5	3.5	V
Total power dissipation		1	W
Maximum current out of all DVss pins		400	mA
Maximum current into all DVdd pins		400	mA
Maximum allowable sink current per I/O pin		160	mA
Maximum allowable source current per I/O pin (excluding port G)		160	mA
Maximum allowable source current per G pin		20	mA
Maximum allowable sink current per group of I/O pins between IOVss pins		160	mA
Maximum allowable source current per group of I/O pins between IOVdd pins (excluding port G)		160	mA
Latchup		200	mA
θ_{JA}, IP2022/PQ80 -120U, airflow = 0 m/sec		46	°C/W
Flash block erase cycle lifetime (if using 20ms block erases - Section 7.1.5)	20K		Cycles
Flash bulk erase cycle lifetime	20K		Cycles
Flash Data Retention, Ta<85C, with 20k cycles typical block/bulk erase cycles	10		years
ESD Human Body Model - all pins	2000		V
ESD Machine Model - all pins	200		V
Max Allowable Junction Temperature		125	°C

Note: To pass system ESD testing for CE Mark, do not connect the Comparator pins (RG0, RG1, RG2) to the Ethernet circuit. Also search for “BREAKX” in the IP2022 silicon errata sheet.

8.2 DC Specifications:

Operating Temperature -40°C ≤ Ta ≤ +85°C

Symbol	Parameter	Min	Typ	Max	Units	Conditions
DVdd	Digital supply voltage	2.3	2.5	2.7	V	
AVdd	Analog supply voltage	2.3	2.5	2.7	V	= DVdd
GVdd	Port G supply voltage	2.3	2.5	2.7	V	= DVdd
XVdd	PLL supply voltage	2.3	2.5	2.7	V	= DVdd
IOVdd	I/O supply voltage (except Port G)	≥ DVdd	2.5 or 3.3	3.6	V	See note 4
Idd	Supply current, full operation DVdd + AVdd + GVdd + XVdd		150		mA	DVdd = 2.3 - 2.7V, 120 MHz CPU core executing 100% of time (during ISR and main program code)
			70		mA	DVdd = 2.5V, 120 MHz CPU core in ISR only, 30MHz CPU core in main program code (Webserver application)
IddIO	Supply current, full operation IOVdd only				mA	IOVdd = DVdd - 3.6V No loads, no floating inputs
Isleep	Supply current, sleep DVdd + AVdd + GVdd + XVdd		200		µA	DVdd = 2.7V, PLL and oscillators off (XCFG bit 4 = 1, CMPCFG bit 7 = 0, ADCCFG bit 3 = 0, XCFG bit 0 = 0, FUSE1 bit 3 = 0)
IsleepIO	Supply current, sleep IOVdd only				µA	IOVdd = 3.6V, PLL and oscillators off, No loads, no floating inputs
Vih	Input high voltage, Port A through Port F	1.8		5.5	V	DVdd = 2.3 - 2.7V, IOVdd = DVdd - 3.6V
	Input high voltage, OSC1 and RTCLK1 inputs	1.8		DVdd	V	
	Input high voltage, $\overline{\text{RST}}$, TSCK, TSI, and $\overline{\text{TSS}}$ inputs	2.25		5.5	V	
Vil	Input low voltage, Port A through Port F			1.0	V	DVdd = 2.3 - 2.7V, IOVdd = DVdd - 3.6V
	Input low voltage, OSC1 and RTCLK1 inputs			0.4	V	
	Input low voltage, $\overline{\text{RST}}$, TSCK, TSI, and $\overline{\text{TSS}}$ inputs			0.9	V	
Vina	Analog input voltage (Port G)			GVdd	V	See note 1
Iil	Input leakage current for Port A through Port G, RTCLK1, and TSO pins	-10	±0.001	10	µA	Port G = 0V or AVdd (see note 1), RTCLK1 = 0V or DVdd All other inputs = 0V or 3.6V TSO measured while $\overline{\text{TSS}}$ = 1 RTCLK1 measured in sleep mode and FUSE0 bit 14 = 1

Symbol	Parameter	Min	Typ	Max	Units	Conditions
Iil	Input leakage current for OSC1 pin	-10		10	µA	OSC1 = 0V or DVdd OSC1 measured in sleep mode and FUSE0 bit 15 = 1
Iil	Input pull-up/down leakage current for Port A through Port F pins	-300	-	100	µA	See note 3. Use < 5.3K ohm pulldown resistors, or < 23K ohm pullup resistors for reliable pullup/pulldown.
Iilt	Input leakage current for $\overline{\text{RST}}$, TSCK, TSI, $\overline{\text{TSS}}$ inputs	-60		10	µA	Vin = 0 to 3.6V. These pins have active internal pull-ups to a diode drop below IOVdd. (60K ohm min., 103K ohm typ., 173K ohm max)
Ioh	Output high current from Port A pins and RE5, RE6, RF1 and RF2 pins	24	60	96	mA	Voh = 2.4V IOVdd = 3.0 to 3.6V
	Output high current from Port B pins and RE4:0, RE7, RF7:3, RF0, and TSO pins	11	24	39	mA	
	Output high current from Port C and Port D pins	8	18	29	mA	
	Output high current from Port G pins	4	12	24	mA	Voh = 1.8V GVdd = 2.3 to 2.7V
Iol	Output low current from Port A pins and RE5, RE6, RF1 and RF2 pins	25	40	50	mA	Vol = 0.4V IOVdd = 3.0 to 3.6V
	Output low current from Port B pins and RE4:0, RE7, RF7:3, RF0, and TSO pins	9	16	24	mA	
	Output low current from Port C and Port D pins	6	11	15	mA	
	Output low current from Port G pins	4	13	24	mA	Vol = 0.4V GVdd = 2.3 to 2.7V

1. If Vref is used for the ADC reference voltage (see Section 5.7.1), then the maximum input voltage on a Port G input is Vref.
2. Data in the Typical (“Typ”) column is at 2.5/3.3V, 25°C unless otherwise stated.
3. The Port A through Port F pins have a weak latch, even when the pin is configured as an input, which drives floating I/O pins to 0V or to a diode drop below DVdd. Some current is required to toggle the state of the latch.
4. If IOVdd rises before DVdd, the IP2022 may drive the I/O pins to IOVdd or GND before DVdd has stabilized. However, there is an internal diode from DVdd to IOVdd, which may be damaged if DVDD exceeds IOVDD by > 0.3V, so DVdd must never exceed IOVdd + 0.3V.
5. Dvdd, AVdd, GVdd, and XVdd supply pins must always all be at the same voltage, but should have filters between supplies.

8.3 AC Specifications:

Operating Temperature: -40°C ≤ Ta ≤ +85°C

Symbol	Parameter	Min	Typ	Max	Units	Conditions
Fcore	CPU core clock frequency	0		120	MHz	Execution from program RAM
Fflash	CPU core clock frequency	0		40	MHz	Execution from program flash
Fsys	System clock frequency	0		120	MHz	
Fosc	External clock frequency on OSC1	0		120	MHz	
Foscr	External clock frequency on RTCLK1	0		120	MHz	
Fxo	Crystal frequency on OSC1, OSC2 pins	4.75	4.8	5.0	MHz	Ext. crystal, ±100 ppm
Fpll	PLL input frequency, after predivider	4.75	4.8	5.0	MHz	TRIM0=FBFE
Fxr	Crystal frequency on RTCLK1, RTCLK2 pins	32.765	32.768	32.771	kHz	Ext. 32.768 KHz crystal, ±100 ppm
Tosl, Tosh	Clock in (OSC1) low or high time	3			ns	
Trl, Trh	Clock in (RTCLK1) low or high time	3.5			ns	
SVdd	DVdd slew rate to ensure Power-On reset	0.05			V/ms	See note 6.
Rpw	$\overline{\text{RST}}$ pin pulse width to insure proper reset, after execution of some code.	5			µs	

6. Vdd must start rising from Vss to ensure proper Power-On-Reset when relying on the internal Power-On-Reset circuitry. If power supply takes more than 50ms to rise from 0 to 2.5V, use RCs on $\overline{\text{RST}}$ pin (see Figure 3-15).

8.4 Comparator DC and AC Specifications

Operating Temperature: –40°C ≤ Ta ≤ +85°C

Parameter	Min	Typ	Max	Units	Conditions
Input offset voltage		±10	±25	mV	CMPT2:0 bits in TRIM0 register are 111
Hysteresis, rising or falling edge	20	50	80	mV	
Bandwidth		15		MHz	min. 100 mV peak-to-peak
Response time			100	ns	Voverdrive = 50 mV Does not include comparator mode entry stabilization time
Time from enabling comparator until output is valid		2000		ns	
Input voltage range	0.1		AVdd - 0.1	V	

8.5 ADC 10-bit Converter DC and AC Specifications

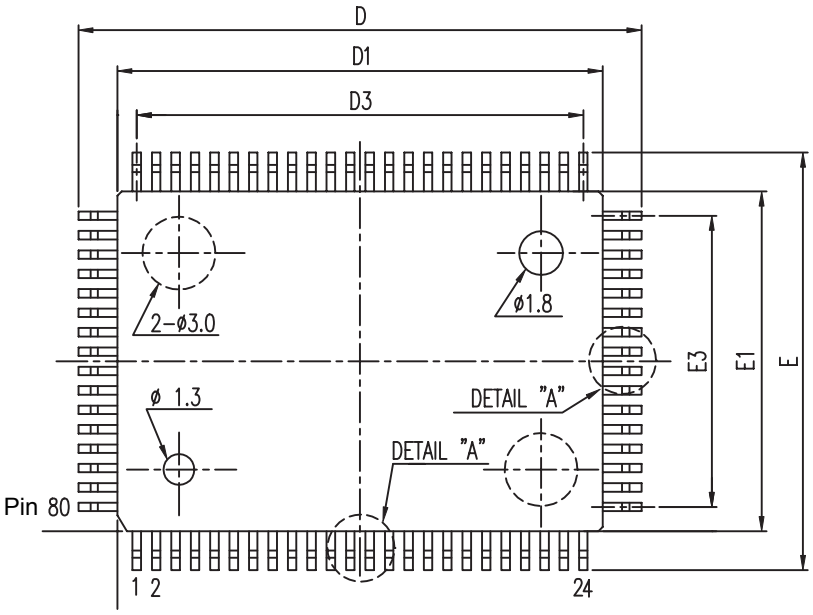
Vref = AVdd, -40°C ≤ Ta ≤ +85°C

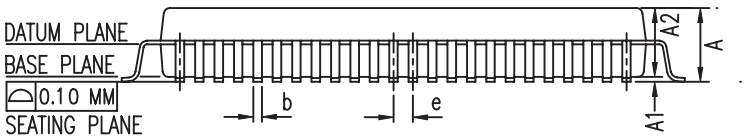
Parameter	Min	Typ	Max	Units	Conditions
Sampling Rate			48	kHz	
Conversion Time			20.8	µs	
Differential nonlinearity error (DNL)			±1.0	LSB	
Integral nonlinearity error (INL)			±1.25	LSB	
Offset error			±1.0	LSB	
Full-scale error			±1.0	LSB	

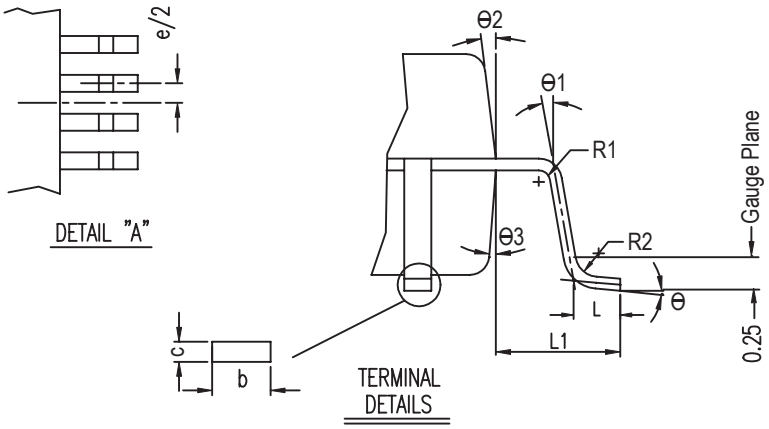
9.0 Package Dimensions

9.1 PQFP

80-pin, 14 mm x 20 mm x 2.72 mm body, 0.8 mm pitch, 17.9 mm x 23.9 mm tip-to-tip. All dimensions in mm.







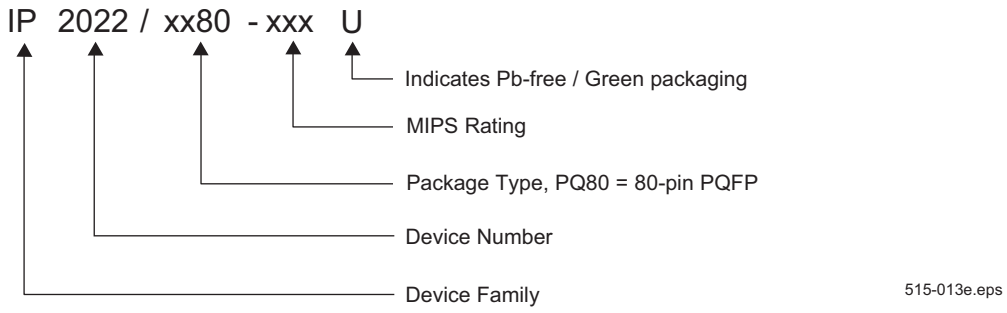
	ASE Package with datecode xxBGyyyyzz, assembled 2004 - 2006			Greatek Package with datecode xxBKyyyyzz, assembled 2006 and later		
	min	typ	max	min	typ	max
D		23.90			23.90	
D1		20.00			20.00	
D3		18.40				
E		17.90			17.90	
E1		14.00			14.00	
E3		12.00				
b	0.30	0.35	0.45	0.30		0.45
c	0.11	0.15	0.23	0.11	0.15	0.23
e		0.80			0.80	
A			3.40			3.40
A1	0.25			0.25		
A2	2.55	2.72	3.05	2.55	2.72	3.05
R1	0.13					
R2	0.13		0.30			
Θ	0°	3.5°	7°	0°	3.5°	7°
Θ1	0°					
Θ2		15°				
Θ3		15°				
L	0.73	0.88	1.03	0.73	0.88	1.03
L1		1.95			1.95	
N	80L			80 L		
JEDEC	MO-112			MO-112		

515-102.eps

10.0 Part Numbering

Table 10-1 Ordering Information

Device	Pins	I/O	Package	Program Flash	Program RAM	Data RAM	Temperature	Max Freq	Package
IP2022/PQ80-120U	80	52	PQFP	64KBytes (32K x 16)	16KBytes (8K x 16)	4KBytes	-40° to 85°C	120MHz	Pb-Free/Green/ RoHS compliant



Sales and Technical Support Contact Information

For the latest contact and support information on IP devices, please visit the Ubicom Web site at www.ubicom.com. The site contains technical literature, local sales contacts, tech support, and many other features.

The Products are not authorized for use in life support systems or under conditions where failure of the Product would endanger the life or safety of the user, except when prior written approval is obtained from Ubicom, Inc. Ask your sales representive for details.



UBICOM™

510 N. Pastoria Ave.
Sunnyvale, CA 94085

Tel: 408-789-2200
Fax: 408-739-2427
Email: sales@ubicom.com
Web: www.ubicom.com

Ubicom develops processors, operating systems and software tools for building home network and media devices. The company provides optimized system-level solutions to OEMs for a wide range of products including wireless routers, access points, bridges, VoIP gateways, digital media devices and other network devices. Ubicom's unique multithreaded processor design, real-time operating system, and application-level solutions combine to ensure a high-quality user experience with unprecedented ease of use. Ubicom has corporate headquarters in Sunnyvale, California, and offices in Taiwan, Turkey, Hong Kong and Netherlands. For more information, visit www.ubicom.com.

Copyright © 2005-2008 Ubicom, Inc. All rights reserved. Ubicom, StreamEngine, UBICOM32, UBICOM16, and Media Meets Networking are trademarks of Ubicom, Inc. All other trademarks are the property of their respective holders.